Package: nichenetr (via r-universe)

May 31, 2026

Type Package

Title Modeling Intercellular Communication by Linking Ligands to Target Genes with NicheNet

Version 2.2.2

Description NicheNet is a computational framework for modeling intercellular communication by linking extracellular ligands to intracellular gene regulatory changes. It integrates prior knowledge of ligand-receptor interactions, signaling pathways, and transcription factor regulation to predict which ligands from sender cells affect gene expression in receiver cells. This enables mechanistic hypothesis generation about cell-cell communication events observed in transcriptomic data.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/Zaoqu-Liu/nichenetr, https://zaoqu-liu.github.io/nichenetr/

BugReports https://github.com/Zaoqu-Liu/nichenetr/issues

RoxygenNote 7.3.3

Depends R (>= 3.0.0)

Imports tidyverse, data.table, dplyr, ggplot2, magrittr, purrr, readr, tibble, tidyr, igraph, Matrix, fdrtool, ROCR, caTools, Hmisc, caret, randomForest, DiagrammeR, mlrMBO, parallelMap, emoa, DiceKriging, e1071, Seurat, cowplot, ggpubr, circlize, ComplexHeatmap, grDevices, ggnewscale, ggforce, shadowtext, Rcpp (>= 1.0.0)

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, RColorBrewer, rmarkdown, testthat, doMC, limma, mco, parallel, covr, sf, qs, digest, data.table, future, future.apply

VignetteBuilder knitr

Config/pak/sysreqs
libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev libglu1-mesa-dev libgmp3-dev make libgsl0-dev libharfbuzz-dev jags libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libmpfr-dev libopenmpi-dev libssl-dev perl libproj-dev python3 libx11-dev zlib1g-dev

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-23 18:38:03 UTC

RemoteUrl https://github.com/Zaoqu-Liu/nichenetr

RemoteRef master

RemoteSha 54c2cb1cc336fe166ee565f6a52cd4ba9fea801c

Contents

`add_hyperparameters_parameter_settings` *Add hyperparameters to existing parameter settings*

Description

`add_hyperparameters_parameter_settings` will generate a list of lists containing the parameter values that need to be used for model construction.

Usage

```
add_hyperparameters_parameter_settings(parameter_setting, lr_sig_hub, gr_hub, ltf_cutoff, algorithm,
```

Arguments

parameter_setting	A list of lists. Sublists contains parameters (like data source weights) and novel sublists will be created by this function by adding additional parameters.
lr_sig_hub	a number between 0 and 1. 0: no correction for hubiness; 1: maximal correction for hubiness.
gr_hub	a number between 0 and 1. 0: no correction for hubiness; 1: maximal correction for hubiness.
ltf_cutoff	Ligand-tf scores beneath the "ltf_cutoff" quantile will be set to 0. Default: 0.99 such that only the 1 percent closest tfs will be considered as possible tfs downstream of the ligand of choice.
algorithm	Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights). Default and recommended: PPR.
damping_factor	Only relevant when algorithm is PPR. In the PPR algorithm, the damping factor is the probability that the random walker will continue its walk on the graph; 1-damping factor is the probability that the walker will return to the seed node. Default: 0.5.
correct_topology	This parameter indicates whether the PPR-constructed ligand-target matrix will be subtracted by a PR-constructed target matrix. TRUE or FALSE.

Value

A list of lists. Every sub-list contains parameter values for a different parameter.

Examples

```
## Not run:
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g

## End(Not run)
```

---

add_ligand_popularity_measures_to_perfs

Merge target gene prediction performances with popularity measures of ligands

---

Description

add_ligand_popularity_measures_to_perfs: Get a data.frame in which the performance measures for target gene prediction of a ligand are merged with the number of times the ligand is mentioned in the Pubmed literature. Serves to investigate popularity bias (i.e. it can be expected that frequenly cited ligands will have better predictive performance because they are better studied).

Usage

```
add_ligand_popularity_measures_to_perfs(performances, ncitations)
```

Arguments

performances	A data.frame in which the performance measures for target gene predictions of ligands are denoted
ncitations	A data frame denoting the number of times a gene is mentioned in the Pubmed literature. Should at least contain following variables: 'symbol' and 'ncitations'. Default: ncitations (variable contained in this package). See function get_ncitations_genes for a function that makes this data frame from current Pubmed information.

Value

A data.frame in which the performance measures for target gene prediction of a ligand are merged with the number of times the ligand is mentioned in the Pubmed literature.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:10], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
performances <- bind_rows(lapply(settings, evaluate_target_prediction, ligand_target_matrix))
# ncitations = get_ncitations_genes()
performances_ligand_popularity <- add_ligand_popularity_measures_to_perfs(performances, ncitations)

## End(Not run)
```

---

add_new_datasource *Add a new data source to the model*

Description

add_new_datasource adds a new data source to one of the ligand-receptor, signaling and gene regulatory data sources.

Usage

```
add_new_datasource(new_source, network, new_weight,source_weights_df)
```

Arguments

new_source	A data frame / tibble containing novel interactions of the ligand-receptor, signaling or gene regulatory layer (required columns: from, to, source)
network	NULL or a data frame / tibble containing the base network to which you want to add the new data source (required columns: from, to, source)
new_weight	a weight value between 0 and 1 to assign to your new data source
source_weights_df	A data frame / tibble containing the weights associated to each already included individual data source (required columns: source, weight).

Value

A list containing 2 elements (network and source_weights_df): the updated network containing your data source and the updated source_weights_df containing the weight of the newly added data source.

Examples

```
## Not run:
## Update the lr_network with a new ligand-receptor data source
library(dplyr)
lr_toy <- tibble(from = "A", to = "B", source = "toy")
new_lr_network <- add_new_datasource(lr_toy, lr_network, 1, source_weights_df)

## End(Not run)
```

---

alias_to_symbol_seurat *Convert aliases to official gene symbols in a Seurat Object*

Description

alias_to_symbol_seurat Convert aliases to official gene symbols in a Seurat Object. Makes use of ‘convert_alias_to_symbols‘

Usage

```
alias_to_symbol_seurat(seurat_obj, organism)
```

Arguments

seurat_obj	Seurat object, v4 or below. For Seurat v5, a warning is thrown and the same object will be returned.
organism	Is Seurat object data from "mouse" or "human"

Value

Seurat object

Examples

```
## Not run:
seurat_object_lite <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj_test.rds"))
seurat_object_lite <- seurat_object_lite %>% alias_to_symbol_seurat("human")

## End(Not run)
```

---

annotation_data_sources

Annotation table of all data sources used in the NicheNet model

---

Description

A data.frame/tibble describing for each type of data source the "parent" database and type of interactions it countains.

Usage

```
annotation_data_sources
```

Format

A data frame/tibble

source Name of the data source

database Name of the comprehensive database to which the data source belongs

type_db Type of data source

type_interaction Interaction type

network Network layer to which the data source belongs: ligand_receptor, signaling or gene_regulatory

apply_hub_corrections *Apply hub corrections to the weighted integrated ligand-signaling and gene regulatory network*

Description

apply_hub_corrections downweighs the importance of nodes with a lot of incoming links in the ligand-signaling and/or gene regulatory network. Hub correction method according to following equation: $Wcor = W * D^{-}h$ with D the indegree matrix of the respective network and h the correction factor.

Usage

```
apply_hub_corrections(weighted_networks, lr_sig_hub, gr_hub)
```

Arguments

weighted_networks	A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
lr_sig_hub	a number between 0 and 1. 0: no correction for hubiness; 1: maximal correction for hubiness.
gr_hub	a number between 0 and 1. 0: no correction for hubiness; 1: maximal correction for hubiness.

Value

A list containing 2 elements (lr_sig and gr): the hubiness-corrected integrated weighted ligand-signaling and gene regulatory networks in data frame / tibble format with columns: from, to, weight.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
wn <- apply_hub_corrections(weighted_networks, lr_sig_hub = 0.5, gr_hub = 0.5)

## End(Not run)
```

assess_influence_source *Assess the influence of an individual data source on ligand-target probability scores*

Description

assess_influence_source will assess the influence of an individual data source on ligand-target probability scores (or rankings of these). Possible output: the ligand-target matrices of the complete model vs the leave-one-out model in which the data source of interest was left out; or a list indicating which target genes for every ligand of interest are affected the most by leaving out the data source of interest.

Usage

```
assess_influence_source(source, lr_network, sig_network, gr_network, source_weights_df, ligands, rank
```

Arguments

source	Name of the data source that will be left out to assess its influence.
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
source_weights_df	A data frame / tibble containing the weights associated to each individual data source. Sources with higher weights will contribute more to the final model performance (required columns: source, weight). Note that only interactions described by sources included here, will be retained during model construction.
ligands	List of ligands for which the model should be constructed
rankings	Indicate whether the output of the models should be the ranking of target gene probability scores (TRUE; top target gene rank = 1) or the scores themselves (FALSE). Default: FALSE.
matrix_output	Indicate whether the output should be the 2 ligand-target matrices (complete model and leave-one-out model) (TRUE) or a listing of genes of which the ligand-target scores/rankings were influenced the most (FALSE). Default: FALSE.
secondary_targets	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

remove_direct_links
: Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

...
: Argumentes for the function add_hyperparameters_parameter_settings

Value

If matrix_output == TRUE: A list of sublists; every sublist contains the elements $model_name and $model: the constructed ligand-target matrix. If matrix_output == FALSE: A list of sublist: every sublist contains; $ligand: name of the ligand tested; $targets_higher: sorted vector of ligand-target scores or rankings of target that score higher in the complete model compared to the leave-one-out model; targets_lower: sorted vector of ligand-target scores or rankings of target that score lower in the complete model compared to the leave-one-out model.

Examples

```
## Not run:
ligands <- extract_ligands_from_settings(expression_settings_validation[1:4])
output <- assess_influence_source("ontogenet", lr_network, sig_network, gr_network, source_weights_df, ligands, l

## End(Not run)
```

---

assess_rf_class_probabilities

Assess probability that a target gene belongs to the geneset based on a multi-ligand random forest model

---

Description

assess_rf_class_probabilities Assess probability that a target gene belongs to the geneset based on a multi-ligand random forest model (with cross-validation). Target genes and background genes will be split in different groups in a stratified way.

Usage

```
assess_rf_class_probabilities(round,folds,geneset,background_expressed_genes,ligands_oi,ligand_targ
```

Arguments

round
: Integer describing which fold of the cross-validation scheme it is.

folds
: Integer describing how many folds should be used.

geneset
: Character vector of the gene symbols of genes of which the expression is potentially affected by ligands from the interacting cell.

background_expressed_genes
: Character vector of gene symbols of the background, non-affected, genes (can contain the symbols of the affected genes as well).

ligands_oi Character vector giving the gene symbols of the ligands you want to build the multi-ligand with.

ligand_target_matrix
The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).

Value

A tibble with columns: $gene, $response, $prediction. Response indicates whether the gene belongs to the geneset of interest, prediction gives the probability this gene belongs to the geneset according to the random forest model.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
geneset <- c("SOCS2", "SOCS3", "IRF1")
background_expressed_genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
fold1_rf_prob <- assess_rf_class_probabilities(round = 1, folds = 2, geneset = geneset, background_expressed_genes

## End(Not run)
```

---

assign_ligands_to_celltype

Assign ligands to cell types

Description

Assign ligands to a sender cell type, based on the strongest expressing cell type of that ligand. Ligands are only assigned to a cell type if that cell type is the only one to show an expression that is higher than the average + SD. Otherwise, it is assigned to "General".

Usage

```
assign_ligands_to_celltype(
  seuratObj,
  ligands,
  celltype_col,
  func.agg = mean,
  func.assign = function(x) {
     mean(x) + sd(x)
 },
  condition_oi = NULL,
  condition_col = NULL,
  ...
)
```

Arguments

seuratObj	Seurat object
ligands	Vector of ligands to assign to cell types
celltype_col	Metadata column name in the Seurat object that contains the cell type information
func.agg	Function to use to aggregate the expression of a ligand across all cells in a cell type (default = mean)
func.assign	Function to use to assign a ligand to a cell type (default = mean + SD)
condition_oi	Condition of interest to subset the Seurat object (default = NULL)
condition_col	Metadata column name in the Seurat object that contains the condition of interest (default = NULL)
...	Arguments passed to Seurat::GetAssayData, e.g., for the slot/layer to use (default: data)

Details

If the provided slot/layer is "data", the normalized counts are first exponentiated before aggregation is performed

Value

A data frame of two columns, the cell type the ligand has been assigned to (`ligand_type`) and the ligand name (`ligand`)

Examples

```
## Not run:
assign_ligands_to_celltype(
  seuratObj = seuratObj, ligands = best_upstream_ligands[1:20],
  celltype_col = "celltype", func.agg = mean, func.assign = function(x) {
    mean(x) + sd(x)
  },
  condition_oi = "LCMV", condition_col = "aggregate", slot = "data"
)

## End(Not run)
```

---

bootstrap_ligand_activity_analysis

Run ligand activity analysis with bootstrap

Description

`bootstrap_ligand_activity_analysis` Randomly sample a gene set from all expressed genes in the receiver cell type, then perform ligand activity analysis on this gene set. This 'null gene set' has equal length to the gene set of interest.

Usage

```
bootstrap_ligand_activity_analysis(expressed_genes_receiver, geneset_oi, background_expressed_genes,
```

Arguments

expressed_genes_receiver	Genes expressed in the receiver cell type
geneset_oi	Character vector of the gene symbols of genes of which the expression is potentially affected by ligands from the interacting cell.
background_expressed_genes	Character vector of gene symbols of the background, non-affected, genes (can contain the symbols of the affected genes as well).
ligand_target_matrix	The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).
potential_ligands	Character vector giving the gene symbols of the potentially active ligands you want to define ligand activities for.
n_iter	Number of iterations to perform (Default: 10)
n_cores	Number of cores to use for parallelization (Default: 1)
parallel_func	Parallelization function to use from "mclapply", "pbmclapply", or "parLapply" (Default: "mclapply")

Value

List of n_iter elements, each element containing the output of predict_ligand_activities for a random gene set

Examples

```
## Not run:
permutations <- bootstrap_ligand_activity_analysis(expressed_genes_receiver, geneset_oi, background_expressed_ge
  ligand_target_matrix, potential_ligands,
  n_iter = 10, n_cores = 1, parallel_func = "mclapply"
)

## End(Not run)
```

---

calculate_de *Calculate differential expression of one cell type versus all other cell types*

Description

calculate_de Calculate differential expression of one cell type versus all other cell types using Seurat::FindAllMarkers. If condition_oi is provided, only consider cells from that condition.

Usage

```
calculate_de(seurat_obj, celltype_colname, condition_oi = NA, condition_colname = NA, assay_oi = "RNA",
```

Arguments

seurat_obj	Seurat object
celltype_colname	Name of the meta data column that indicates the cell type of a cell
condition_oi	If provided, subset seurat_obj so DE is only calculated for cells belonging to condition_oi
condition_colname	Name of the meta data column that indicates from which group/condition a cell comes from
assay_oi	Which assay need to be used for DE calculation. If NULL, will use DefaultAssay
...	Arguments passed to Seurat::FindAllMarkers(by default: features = NULL, min.pct = 0, logfc.threshold = 0, return.thresh = 1)

Value

A dataframe containing the DE results

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj.rds"))
seurat_obj$celltype <- make.names(seurat_obj$celltype)
# Calculate cell-type specific markers across conditions
calculate_de(seurat_obj, "celltype")
# Calculate LCMV-specific cell-type markers
calculate_de(seurat_obj, "celltype", condition_oi = "LCMV", condition_colname = "aggregate")

## End(Not run)
```

---

calculate_fraction_top_predicted

Determine the fraction of genes belonging to the geneset or background and to the top-predicted genes.

Description

calculate_fraction_top_predicted Defines the fraction of genes belonging to the geneset or background and to the top-predicted genes.

Usage

```
calculate_fraction_top_predicted(affected_gene_predictions, quantile_cutoff = 0.95)
```

Arguments

affected_gene_predictions
: Tibble with columns "gene", "prediction" and "response" (e.g. output of function ‘assess_rf_class_probabilities‘)

quantile_cutoff
: Quantile of which genes should be considered as top-predicted targets. Default: 0.95, thus considering the top 5 percent predicted genes as predicted targets.

Value

A tibble indicating the number of genes belonging to the gene set of interest or background (true_target column), the number and fraction of genes of these gruops that were part of the top predicted targets in a specific cross-validation round.

---

calculate_fraction_top_predicted_fisher

Perform a Fisher's exact test to determine whether genes belonging to the gene set of interest are more likely to be part of the top-predicted targets.

Description

`calculate_fraction_top_predicted_fisher` Performs a Fisher's exact test to determine whether genes belonging to the gene set of interest are more likely to be part of the top-predicted targets.

Usage

```
calculate_fraction_top_predicted_fisher(affected_gene_predictions, quantile_cutoff = 0.95, p_value_ou
```

Arguments

affected_gene_predictions
: Tibble with columns "gene", "prediction" and "response" (e.g. output of function ‘assess_rf_class_probabilities‘)

quantile_cutoff
: Quantile of which genes should be considered as top-predicted targets. Default: 0.95, thus considering the top 5 percent predicted genes as predicted targets.

p_value_output
: Should total summary or p-value be returned as output? Default: TRUE.

Value

Summary of the Fisher's exact test or just the p-value

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
geneset <- c("SOCS2", "SOCS3", "IRF1")
background_expressed_genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
gene_predictions_list <- seq(2) %>% lapply(assess_rf_class_probabilities, 2, geneset = geneset, background_expres
target_prediction_performances_fisher_pval <- gene_predictions_list %>%
  lapply(calculate_fraction_top_predicted_fisher) %>%
  unlist() %>%
  mean()

## End(Not run)
```

---

calculate_niche_de *Calculate differential expression of cell types in one niche versus all other niches of interest.*

Description

calculate_niche_de Calculate differential expression of cell types in one niche versus all other niches of interest. This is possible for sender cell types and receiver cell types.

Usage

```
calculate_niche_de(seurat_obj, niches, type, assay_oi = "SCT")
```

Arguments

seurat_obj	Seurat object
niches	a list of lists/niches giving the name, senders and receiver celltypes for each nice. Sender and receiver cell types should be part of Idents(seurat_obj).
type	For what type of cellype is the DE analysis: "sender" or "receiver"?
assay_oi	Which assay need to be used for DE calculation via 'FindMarkers'. Default SCT, alternatives: RNA.

Value

A tibble containing the DE results of the niches versus each other.

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
calculate_niche_de(seurat_obj, niches, "sender")

## End(Not run)
```

---

calculate_niche_de_targets

Calculate differential expression of receiver cell type in one niche versus all other niches of interest: focus on finding DE genes

---

Description

`calculate_niche_de_targets` Calculate differential expression of receiver cell type in one niche versus all other niches of interest: focus on finding DE genes

Usage

```
calculate_niche_de_targets(seurat_obj, niches, expression_pct, lfc_cutoff, assay_oi = "SCT")
```

Arguments

`seurat_obj`	Seurat object
`niches`	a list of lists/niches giving the name, senders and receiver celltypes for each nice. Sender and receiver cell types should be part of Idents(seurat_obj).
`expression_pct`	input of 'min.pct' of 'Seurat::FindMarkers'
`lfc_cutoff`	input of 'logfc.threshold' of 'Seurat::FindMarkers'
`assay_oi`	Which assay need to be used for DE calculation via 'FindMarkers'. Default SCT, alternatives: RNA.

Value

A tibble containing the DE results of the niches versus each other.

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
calculate_niche_de_targets(seurat_obj, niches, expression_pct = 0.10, lfc_cutoff = 0.15)

## End(Not run)
```

---

calculate_p_value_bootstrap

Calculate ligand p-values from the bootstrapped ligand activity analysis

---

Description

`calculate_p_value_bootstrap` Calculate the p-value for each ligand from the bootstrapped ligand activity analysis

Usage

```
calculate_p_value_bootstrap(bootstrap_results, ligand_activities, metric = "aupr_corrected")
```

Arguments

`bootstrap_results`	Output of `bootstrap_ligand_activity_analysis`
`ligand_activities`	Output of `predict_ligand_activities`
`metric`	Metric to use (Default: "aupr_corrected")

Value

Data frame with the ligand name, the number of times the bootstrapped value had a higher score than the observed (`total`), and the p-value for each ligand, calculated as `total/length(bootstrap_results)`

Examples

```
## Not run:
permutations <- bootstrap_ligand_activity_analysis(expressed_genes_receiver, geneset_oi, background_expressed_ge
  ligand_target_matrix, potential_ligands,
  n_iter = 10
)
p_values <- calculate_p_value_bootstrap(permutations, ligand_activities, metric = "aupr_corrected")

## End(Not run)
```

---

calculate_spatial_DE *Calculate differential expression between spatially different subpopulations of the same cell type*

Description

`calculate_spatial_DE` Calculate differential expression between spatially different subpopulations of the same cell type

Usage

```
calculate_spatial_DE(seurat_obj, spatial_info, assay_oi = "SCT")
```

Arguments

`seurat_obj`	Seurat object
`spatial_info`	Tibble giving information about which celltypes should be compared to each other for defining spatial differential expression. Contains the columns "celltype_region_oi", "celltype_other_region", "niche", "celltype_type".
`assay_oi`	Assay for the DE analysis: RNA, SCT, ...

Value

A tibble with DE output

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
spatial_info <- tibble(
  celltype_region_oi = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
 celltype_other_region = c("LSECs_central", "Hepatocytes_central", "Stellate cells_central")
) %>%
```

```
  mutate(niche = "KC_niche", celltype_type = "sender")
calculate_spatial_DE(seurat_obj, spatial_info)

## End(Not run)
```

`classification_evaluation_continuous_pred_wrapper` *Assess how well classification predictions accord to the expected response*

Description

`classification_evaluation_continuous_pred_wrapper` Assess how well classification predictions accord to the expected response.

Usage

```
classification_evaluation_continuous_pred_wrapper(response_prediction_tibble)
```

Arguments

`response_prediction_tibble`
: Tibble with columns "response" and "prediction" (e.g. output of function 'assess_rf_class_probabilities')

Value

A tibble showing several classification evaluation metrics.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
geneset <- c("SOCS2", "SOCS3", "IRF1")
background_expressed_genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
fold1_rf_prob <- assess_rf_class_probabilities(round = 1, folds = 2, geneset = geneset, background_expressed_genes
classification_evaluation_continuous_pred_wrapper(fold1_rf_prob)

## End(Not run)
```

clear_database_cache *Clear Database Cache*

Description

Removes all cached databases from memory.

Usage

```
clear_database_cache()
```

combine_sender_receiver_de *Combine the differential expression information of ligands in the sender celltypes with the differential expression information of their cognate receptors in the receiver cell types*

Description

combine_sender_receiver_de Combine the differential expression information of ligands in the sender celltypes with the differential expression information of their cognate receptors in the receiver cell types.

Usage

```
combine_sender_receiver_de(DE_sender_processed, DE_receiver_processed, lr_network, specificity_score
```

Arguments

DE_sender_processed	Output of 'process_niche_de' with 'type = receiver'
DE_receiver_processed	Output of 'process_niche_de' with 'type = receiver'
lr_network	Ligand-Receptor Network in tibble format: ligand, receptor as columns
specificity_score	Defines which score will be used to prioritze ligand-receptor pairs and consider their differential expression. Default and recommended: "min_lfc". "min_lfc" looks at the minimal logFC of the ligand/receptor between the celltype of interest and all the other celltypes. Alternatives: "mean_lfc", "min_score", and "mean_score". Mean uses the average/mean instead of minimum. score is the product of the logFC and the ratio of fraction of expressing cells.

Value

A tibble giving the differential expression information of ligands in the sender celltypes and their cognate receptors in the receiver cell types.

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
DE_sender <- calculate_niche_de(seurat_obj, niches, "sender")
DE_receiver <- calculate_niche_de(seurat_obj, niches, "receiver")
expression_pct <- 0.10
DE_sender_processed <- process_niche_de(DE_table = DE_sender, niches = niches, expression_pct = expression_pct, ty
DE_receiver_processed <- process_niche_de(DE_table = DE_receiver, niches = niches, expression_pct = expression_pct
specificity_score_LR_pairs <- "min_lfc"
DE_sender_receiver <- combine_sender_receiver_de(DE_sender_processed, DE_receiver_processed, lr_network, specifi

## End(Not run)
```

---

construct_ligand_target_matrix

Construct a ligand-target probability matrix for ligands of interest.

Description

construct_ligand_target_matrix Convert integrated weighted networks into a matrix containg ligand-target probability scores. The higher this score, the more likely a particular ligand can induce the expression of a particular target gene.

Usage

```
construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorithm =
```

Arguments

weighted_networks	A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)

`lr_network` A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)

`ligands` A list of all ligands and ligand-combinations of which target gene probability scores should be calculated. Example format: list("TNF","BMP2",c("IL4","IL13")).

`ltf_cutoff` Ligand-tf scores beneath the "ltf_cutoff" quantile will be set to 0. Default: 0.99 such that only the 1 percent closest tfs will be considered as possible tfs downstream of the ligand of choice.

`algorithm` Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights). Default and recommended: PPR.

`damping_factor` Only relevant when algorithm is PPR. In the PPR algorithm, the damping factor is the probability that the random walker will continue its walk on the graph; 1-damping factor is the probability that the walker will return to the seed node. Default: 0.5.

`secondary_targets`
Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

`ligands_as_cols`
Indicate whether ligands should be in columns of the matrix and target genes in rows or vice versa. Default: TRUE

`remove_direct_links`
Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

Value

A matrix containing ligand-target probability scores.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99,

## End(Not run)
```

construct_ligand_tf_matrix *Construct a ligand-tf signaling probability matrix for ligands of interest.*

Description

construct_ligand_tf_matrix Convert integrated weighted networks into a matrix containg ligand-tf probability scores. The higher this score, the more likely a particular ligand can signal to a downstream gene.

Usage

```
construct_ligand_tf_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorithm = "PPR
```

Arguments

weighted_networks	A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
ligands	A list of all ligands and ligand-combinations of which target gene probability scores should be calculated. Example format: list("TNF","BMP2",c("IL4","IL13")).
ltf_cutoff	Ligand-tf scores beneath the "ltf_cutoff" quantile will be set to 0. Default: 0.99 such that only the 1 percent closest tfs will be considered as possible tfs downstream of the ligand of choice.
algorithm	Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights + give the ligand itself the max score). Default and recommended: PPR.
damping_factor	Only relevant when algorithm is PPR. In the PPR algorithm, the damping factor is the probability that the random walker will continue its walk on the graph; 1-damping factor is the probability that the walker will return to the seed node. Default: 0.5.
ligands_as_cols	Indicate whether ligands should be in columns of the matrix and target genes in rows or vice versa. Default: FALSE

Value

A matrix containing ligand-target probability scores.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_tf <- construct_ligand_tf_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorithm = "PR

## End(Not run)
```

`construct_model` *Construct a ligand-target model given input parameters.*

Description

`construct_model` will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix.

Usage

```
construct_model(parameters_setting, lr_network, sig_network, gr_network, ligands, secondary_targets =
```

Arguments

`parameters_setting`	A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See `prepare_settings_leave_o` and `add_hyperparameters_parameter_settings`.
`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)
`gr_network`	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
`ligands`	List of ligands for which the model should be constructed
`secondary_targets`	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE
`remove_direct_links`	Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

Value

A list containing following elements: $model_name and $model.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
ligands <- extract_ligands_from_settings(settings)
models_characterization <- parallel::mclapply(weights_settings_loi[1:3], construct_model, lr_network, sig_networ

## End(Not run)
```

---

construct_random_model

Construct a randomised ligand-target model given input parameters.

Description

`construct_random_model` will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix after randomization by edge swapping.

Usage

```
construct_random_model(parameters_setting, lr_network, sig_network, gr_network, ligands, secondary_ta
```

Arguments

`parameters_setting`	A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See `prepare_settings_leave_o` and `add_hyperparameters_parameter_settings`.
`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)
`gr_network`	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
`ligands`	List of ligands for which the model should be constructed

secondary_targets
: Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

remove_direct_links
: Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

Value

A list containing following elements: $model_name and $model.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
ligands <- extract_ligands_from_settings(settings)
models_characterization <- parallel::mclapply(weights_settings_loi[1:3], construct_random_model, lr_network, sig

## End(Not run)
```

---

construct_tf_target_matrix

Construct a tf-target matrix.

Description

construct_tf_target_matrix Convert integrated gene regulatory weighted network into matrix format.

Usage

```
construct_tf_target_matrix(weighted_networks, tfs_as_cols = FALSE, standalone_output = FALSE)
```

Arguments

weighted_networks
: A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)

tfs_as_cols Indicate whether ligands should be in columns of the matrix and target genes in rows or vice versa. Default: FALSE

standalone_output
Indicate whether the ligand-tf matrix should be formatted in a way convenient to use alone (with gene symbols as row/colnames). Default: FALSE

Value

A matrix containing tf-target regulatory weights.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
tf_target <- construct_tf_target_matrix(weighted_networks, tfs_as_cols = TRUE, standalone_output = TRUE)

## End(Not run)
```

---

construct_weighted_networks

Construct weighted layer-specific networks

Description

construct_weighted_networks construct layer-specific weighted integrated networks from input source networks via weighted aggregation.

Usage

```
construct_weighted_networks(lr_network, sig_network, gr_network,source_weights_df, n_output_networks
```

Arguments

lr_network A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)

sig_network A data frame / tibble containing signaling interactions (required columns: from, to, source)

gr_network A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)

source_weights_df
A data frame / tibble containing the weights associated to each individual data source. Sources with higher weights will contribute more to the final model performance (required columns: source, weight). Note that only interactions described by sources included here, will be retained during model construction.

n_output_networks
The number of output networks to return: 2 (ligand-signaling and gene regulatory; default) or 3 (ligand-receptor, signaling and gene regulatory).

Value

A list containing 2 elements (lr_sig and gr) or 3 elements (lr, sig, gr): the integrated weighted ligand-signaling and gene regulatory networks or ligand-receptor, signaling and gene regulatory networks in data frame / tibble format with columns: from, to, weight.

Examples

```
## Not run:
## Generate the weighted networks from input source networks
wn <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)

## End(Not run)
```

`convert_alias_to_symbols` *Convert aliases to official gene symbols*

Description

`convert_alias_to_symbols` Convert aliases to offcial gene symbols

Usage

```
convert_alias_to_symbols(aliases, organism, verbose = TRUE)
```

Arguments

`aliases`	A character vector of symbols/aliases
`organism`	"mouse" or "human"
`verbose`	TRUE or FALSE

Value

A character vector of official gene symbols.

Examples

```
library(dplyr)
human_symbols <- c("TNF", "IFNG", "IL-15")
aliases <- human_symbols %>% convert_alias_to_symbols(organism = "human")
```

convert_cluster_to_settings *Convert cluster assignment to settings format suitable for target gene prediction.*

Description

convert_cluster_to_settings Convert cluster assignment to settings format suitable for target gene prediction.

Usage

```
convert_cluster_to_settings(i, cluster_vector, setting_name, setting_from, background = NULL)
```

Arguments

i	The cluster number of the cluster of interest to which genes should belong
cluster_vector	Named vector containing the cluster number to which every gene belongs
setting_name	Base name of the setting
setting_from	Active ligands for the specific setting
background	NULL or a character vector of genes belonging to the background. When NULL: the background will be formed by genes belonging to other clusters that the cluster of interest. Default NULL. If not NULL and genes present in the cluster of interest are in this vector of background gene names, these genes will be removed from the background.

Value

A list with following elements: $name (indicating the cluster id), $from, $response. $response is a gene-named logical vector indicating whether the gene is part of the respective cluster.

Examples

```
## Not run:
genes_clusters <- c("TGFB1" = 1, "TGFB2" = 1, "TGFB3" = 2)
cluster_settings <- lapply(seq(length(unique(genes_clusters))), convert_cluster_to_settings, cluster_vector = ge

## End(Not run)
```

convert_expression_settings_evaluation *Convert expression settings to correct settings format for evaluation of target gene prediction.*

Description

convert_expression_settings_evaluation Converts expression settings to correct settings format for evaluation of target gene prediction.

Usage

```
convert_expression_settings_evaluation(setting)
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$diffexp: data frame or tibble containing at least 3 variables= $gene, $lfc (log fold change treated vs untreated) and $qval (fdr-corrected p-value)

Value

A list with following elements: $name, $from, $response. $response will be a gene-named logical vector indicating whether the gene's transcription was influenced by the active ligand(s) in the setting of interest.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)

## End(Not run)
```

convert_expression_settings_evaluation_regression *Convert expression settings to correct settings format for evaluation of target gene log fold change prediction (regression).*

Description

convert_expression_settings_evaluation_regression Converts expression settings to correct settings format for evaluation of target gene log fold change prediction (regression).

Usage

```
convert_expression_settings_evaluation_regression(setting)
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$diffexp: data frame or tibble containing at least 3 variables= $gene, $lfc (log fold change treated vs untreated) and $qval (fdr-corrected p-value)

Value

A list with following elements: $name, $from, $response. $response will be a gene-named numeric vector of log fold change values.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation_regression)

## End(Not run)
```

---

convert_gene_list_settings_evaluation

Convert gene list to correct settings format for evaluation of target gene prediction.

Description

convert_gene_list_settings_evaluation Converts a gene list to correct settings format for evaluation of target gene prediction.

Usage

```
convert_gene_list_settings_evaluation(gene_list, name, ligands_oi, background)
```

Arguments

gene_list	A character vector of target gene names
name	The name that will be given to the setting
ligands_oi	The possibly active ligands
background	A character vector of names of genes that are not target genes. If genes present in the gene list are in this vector of background gene names, these genes will be removed from the background.

Value

A list with following elements: $name, $from, $response

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
all_genes <- unique(c(weighted_networks$gr$from, weighted_networks$gr$to, weighted_networks$lr_sig$from, weighte
gene_list <- c("ID1", "ID2", "ID3")
setting <- list(convert_gene_list_settings_evaluation(gene_list = c("ID1", "ID2", "ID3"), name = "test", ligands_c

## End(Not run)
```

---

convert_human_to_mouse_symbols

Convert human gene symbols to their mouse one-to-one orthologs.

Description

convert_human_to_mouse_symbols Convert human gene symbols to their mouse one-to-one orthologs.

Usage

```
convert_human_to_mouse_symbols(symbols, version = 1)
```

Arguments

symbols	A character vector of official human gene symbols
version	Indicates which version of the mouse-human annotations should be used: Default 1. In April 2022, and nichenetr 2.0., the default will change to 2.

Value

A character vector of official mouse gene symbols (one-to-one orthologs of the input human gene symbols).

Examples

```
library(dplyr)
human_symbols <- c("TNF", "IFNG")
mouse_symbols <- human_symbols %>% convert_human_to_mouse_symbols()
```

convert_mouse_to_human_symbols *Convert mouse gene symbols to their human one-to-one orthologs.*

Description

convert_mouse_to_human_symbols Convert mouse gene symbols to their human one-to-one orthologs.

Usage

```
convert_mouse_to_human_symbols(symbols, version = 1)
```

Arguments

symbols	A character vector of official mouse gene symbols
version	Indicates which version of the mouse-human annotations should be used: Default 1. In April 2022, and nichenetr 2.0., the default will change to 2.

Value

A character vector of official human gene symbols (one-to-one orthologs of the input mouse gene symbols).

Examples

```
library(dplyr)
mouse_symbols <- c("Tnf", "Ifng")
human_symbols <- mouse_symbols %>% convert_mouse_to_human_symbols()
```

convert_settings_ligand_prediction *Convert settings to correct settings format for ligand prediction.*

Description

convert_settings_ligand_prediction Converts settings to correct settings format for ligand activity prediction. In this prediction problem, ligands (out of a set of possibly active ligands) will be ranked based on feature importance scores. The format can be made suited for: 1) validation of ligand activity state prediction by calculating individual feature importane scores or 2) feature importance based on models with embedded feature importance determination; applications in which ligands need to be scores based on their possible upstream activity: 3) by calculating individual feature importane scores or 4) feature importance based on models with embedded feature importance determination.

Usage

```
convert_settings_ligand_prediction(settings, all_ligands, validation = TRUE, single = TRUE)
```

Arguments

settings	A list of lists. Eeach sublist contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest.
all_ligands	A character vector of possible ligands that will be considered for the ligand activity state prediction.
validation	TRUE if seetings need to be prepared for validation of ligand activity state predictions (this implies that the true active ligand of a setting is known); FALSE for application purposes when the true active ligand(s) is/are not known.
single	TRUE if feature importance scores for ligands will be calculated by looking at ligans individually. FALSE if the goal is to calculate the feature importance scores via sophisticated classification algorithms like random forest.

Value

A list with following elements: $name, $ligand: name of active ligand(s) (only if validation is TRUE), $from (ligand(s) that will be tested for activity prediction), $response

Examples

```
## Not run:
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
ligands <- unlist(extract_ligands_from_settings(settings, combination = FALSE))
settings_ligand_pred <- convert_settings_ligand_prediction(settings, ligands, validation = TRUE, single = TRUE)

## End(Not run)
```

---

convert_settings_tf_prediction

Convert settings to correct settings format for TF prediction.

Description

convert_settings_tf_prediction Converts settings to correct settings format for TF activity prediction. In this prediction problem, TFs (out of a set of possibly active TFs) will be ranked based on feature importance scores. The format can be made suited for applications in which TFs need to be scored based on their possible upstream activity: 3) by calculating individual feature importane scores or 4) feature importance based on models with embedded feature importance determination. Remark that upstream regulator analysis for TFs here is experimental and was not thoroughly validated in the study accompanying this package.

Usage

```
convert_settings_tf_prediction(settings, all_tfs, single = TRUE)
```

Arguments

settings	A list of lists. Eeach sublist contains the following elements: .$name: name of the setting; .$from: name(s) of the tf(s) active in the setting of interest; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest.
all_tfs	A character vector of possible tfs that will be considered for the tf activity state prediction.
single	TRUE if feature importance scores for tfs will be calculated by looking at ligans individually. FALSE if the goal is to calculate the feature importance scores via sophisticated classification algorithms like random forest.

Value

A list with following elements: $name, $tf: name of active tf(s) (only if validation is TRUE), $from (tf(s) that will be tested for activity prediction), $response

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_tf_pred <- convert_settings_tf_prediction(settings, all_tfs = c("SMAD1", "STAT1", "RELA"), single = TRUE)
# show how this function can be used to predict activities of TFs
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
tf_target <- construct_tf_target_matrix(weighted_networks, tfs_as_cols = TRUE, standalone_output = TRUE)
tf_importances <- dplyr::bind_rows(lapply(settings_tf_pred, get_single_ligand_importances, tf_target, known = FAL
print(head(tf_importances))

## End(Not run)
```

---

convert_settings_topn_ligand_prediction

Converts expression settings to format in which the total number of potential ligands is reduced up to n top-predicted active ligands.

Description

convert_expression_settings_evaluation Converts expression settings to format in which the total number of potential ligands is reduced up to n top-predicted active ligands.(useful for applications when a lot of ligands are potentially active, a lot of settings need to be predicted and a multi-ligand model is trained).

Usage

```
convert_settings_topn_ligand_prediction(setting, importances, model, n, normalization)
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$diffexp: data frame or tibble containing at least 3 variables= $gene, $lfc (log fold change treated vs untreated) and $qval (fdr-corrected p-value)
importances	A data frame containing at least folowing variables: $setting, $test_ligand, $ligand and one or more feature importance scores. $test_ligand denotes the name of a possibly active ligand, $ligand the name of the truely active ligand.
model	A model object of a classification object as e.g. generated via caret.
n	The top n number of ligands according to the ligand activity state prediction model will be considered as potential ligand for the generation of a new setting.
normalization	Way of normalization of the importance measures: "mean" (classifcal z-score) or "median" (modified z-score)

Value

A list with following elements: $name, $from, $response. $response will be a gene-named logical vector indicating whether the gene's transcription was influenced by the active ligand(s) in the setting of interest.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
evaluation <- evaluate_importances_ligand_prediction(ligand_importances, "median", "lda")

settings <- lapply(expression_settings_validation[5:10], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
settings <- lapply(settings, convert_settings_topn_ligand_prediction, importances = ligand_importances, model = e

## End(Not run)
```

---

convert_single_cell_expression_to_settings

Prepare single-cell expression data to perform ligand activity analysis

---

Description

convert_single_cell_expression_to_settings Prepare single-cell expression data to perform ligand activity analysis

Usage

```
convert_single_cell_expression_to_settings(cell_id, expression_matrix, setting_name, setting_from, re
```

Arguments

cell_id	Identity of the cell of interest
expression_matrix	Gene expression matrix of single-cells
setting_name	Name of the dataset
setting_from	Character vector giving the gene symbols of the potentially active ligands you want to define ligand activities for.
regression	Perform regression-based ligand activity analysis (TRUE) or classification-based ligand activity analysis (FALSE) by considering the genes expressed higher than the 0.975 quantiles as genes of interest. Default: FALSE.

Value

A list with slots $name, $from and $response respectively containing the setting name, potentially active ligands and the response to predict (whether genes belong to gene set of interest; i.e. most strongly expressed genes in a cell)

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
cell_ids <- c("cell1", "cell2")
expression_scaled <- matrix(rnorm(length(genes) * 2, sd = 0.5, mean = 0.5), nrow = 2)
rownames(expression_scaled) <- cell_ids
colnames(expression_scaled) <- genes
settings <- convert_single_cell_expression_to_settings(cell_id = cell_ids[1], expression_matrix = expression_scal

## End(Not run)
```

correct_topology_ppr *Adapt a ligand-target probability matrix construced via PPR by correcting for network topolgoy.*

Description

correct_topology_ppr The ligand-target probability scores of a matrix constructed via personalized pagerank will be subtracted by target probability scores calculated via global pagerank; these latter scores can be considered as scores solely attributed to network topology and not by proximity to the ligand of interest. Recommended to use this function in combination with a ligand-target matrix constructed without applying a cutoff on the ligand-tf matrix.

Usage

```
correct_topology_ppr(ligand_target_matrix,weighted_networks,ligands_position = "cols")
```

Arguments

ligand_target_matrix	A matrix of ligand-target probabilty scores.
weighted_networks	A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A matrix containing ligand-target probability scores, after subtracting target scores solely due to network topology.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
ligand_target_matrix <- correct_topology_ppr(ligand_target_matrix, weighted_networks, ligands_position = "cols")

## End(Not run)
```

diagrammer_format_signaling_graph *Prepare extracted ligand-target signaling network for visualization with DiagrammeR.*

Description

diagrammer_format_signaling_graph Prepare extracted ligand-target signaling network for visualization with DiagrammeR.

Usage

```
diagrammer_format_signaling_graph(signaling_graph_list, ligands_all,targets_all, sig_color = "steelbl
```

Arguments

signaling_graph_list	A list of two elements: sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
ligands_all	A character vector of one or more ligands of interest
targets_all	A character vector of one or more target genes of interest
sig_color	The color for ligand-signaling edges and the ligand node. Default: steelblue.
gr_color	The color for the gene regulatory edges and the target node. Default: orange.

Value

A DiagrammeR Graph object ready for visualization via DiagrammeR::render_graph.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_tf_matrix <- construct_ligand_tf_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorit
all_ligands <- c("BMP2")
all_targets <- c("HEY1")
top_n_regulators <- 2
ligand_target_signaling_list <- get_ligand_signaling_path(ligand_tf_matrix, all_ligands, all_targets, top_n_regu
graph <- diagrammer_format_signaling_graph(ligand_target_signaling_list, all_ligands, all_targets)
# DiagrammeR::render_graph(graph, layout = "tree")

## End(Not run)
```

estimate_source_weights_characterization
Estimate data source weights of data sources of interest based on leave-one-in and leave-one-out characterization performances.

Description

estimate_source_weights_characterization will estimate data source weights of data sources of interest based on a model that was trained to predict weights of data sources based on leave-one-in and leave-one-out characterization performances.

Usage

```
estimate_source_weights_characterization(loi_performances,loo_performances,source_weights_df, source
```

Arguments

loi_performances	Performances of models in which a particular data source of interest was the only data source in or the ligand-signaling or the gene regulatory network.
loo_performances	Performances of models in which a particular data source of interest was removed from the ligand-signaling or the gene regulatory network before model construction.
source_weights_df	A data frame / tibble containing the weights associated to each individual data source. Sources with higher weights will contribute more to the final model performance (required columns: source, weight). Note that only interactions described by sources included here, will be retained during model construction.
sources_oi	The names of the data sources of which data source weights should be estimated based on leave-one-in and leave-one-out performances.
random_forest	Indicate whether for the regression between leave-one-in + leave-one-out performances and data source weights a random forest model should be trained (TRUE) or a linear model (FALSE). Default: FALSE

Value

A list containing two elements. $source_weights_df (the input source_weights_df extended by the estimated source_weighs for data sources of interest) and $model (model object of the regression between leave-one-in, leave-one-out performances and data source weights).

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network = lr_network, sig_network = sig
```

```
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 4)
job_characterization_loi <- parallel::mclapply(weights_settings_loi[1:4], evaluate_model, lr_network = lr_networ
loi_performances <- process_characterization_target_prediction_average(job_characterization_loi)
weights_settings_loo <- prepare_settings_leave_one_out_characterization(lr_network = lr_network, sig_network = si
weights_settings_loo <- lapply(weights_settings_loo, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 4)
job_characterization_loo <- parallel::mclapply(weights_settings_loo[1:4], evaluate_model, lr_network = lr_networ
loo_performances <- process_characterization_target_prediction_average(job_characterization_loo)
sources_oi <- c("kegg_cytokines")
output <- estimate_source_weights_characterization(loi_performances, loo_performances, source_weights_df %>% fil

## End(Not run)
```

`evaluate_importances_ligand_prediction`
Evaluation of ligand activity prediction based on ligand importance scores.

Description

`evaluate_importances_ligand_prediction` Evaluate how well a trained model of ligand importance scores is able to predict the true activity state of a ligand. For this it is assumed, that ligand importance measures for truely active ligands will be higher than for non-active ligands. A classificiation algorithm chosen by the user is trained to construct one model based on the ligand importance scores of all ligands of interest (ligands importance scores are considered as features). Several classification evaluation metrics for the prediction are calculated and variable importance scores can be extracted to rank the different importance measures in order of importance for ligand activity state prediction.

Usage

```
evaluate_importances_ligand_prediction(importances, normalization, algorithm, var_imps = TRUE, cv = TR
```

Arguments

`importances`	A data frame containing at least folowing variables: $setting, $test_ligand, $ligand and one or more feature importance scores. $test_ligand denotes the name of a possibly active ligand, $ligand the name of the truely active ligand.
`normalization`	Way of normalization of the importance measures: "mean" (classifcal z-score) or "median" (modified z-score)
`algorithm`	The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: with embedded feature selection: "rf","glm","fda","glmnet","sdwd","gam","glmboost"; without: "lda","naive_bayes","pls"(b bug in current version of pls package), "pcaNNet". Please notice that not all these algorithms work when the features (i.e. ligand vectors) are categorical (i.e. discrete class assignments).

var_imps	Indicate whether in addition to classification evaluation performances, variable importances should be calculated. Default: TRUE.
cv	Indicate whether model training and hyperparameter optimization should be done via cross-validation. Default: TRUE. FALSE might be useful for applications only requiring variable importance, or when final model is not expected to be extremely overfit.
cv_number	The number of folds for the cross-validation scheme: Default: 4; only relevant when cv == TRUE.
cv_repeats	The number of repeats during cross-validation. Default: 2; only relevant when cv == TRUE.
parallel	Indiciate whether the model training will occur parallelized. Default: FALSE. TRUE only possible for non-windows OS.
n_cores	The number of cores used for parallelized model training via cross-validation. Default: 4. Only relevant on non-windows OS.
ignore_errors	Indiciate whether errors during model training by caret should be ignored such that another model training try will be initiated until model is trained without raising errors. Default: FALSE.

Value

A list with the following elements. $performances: data frame containing classification evaluation measure for classification on the test folds during training via cross-validation; $performances_training: data frame containing classification evaluation measures for classification of the final model (discrete class assignments) on the complete data set (performance can be severly optimistic due to overfitting!); $performance_training_continuous: data frame containing classification evaluation measures for classification of the final model (class probability scores) on the complete data set (performance can be severly optimistic due to overfitting!) $var_imps: data frame containing the variable importances of the different ligands (embbed importance score for some classification algorithms, otherwise just the auroc); $prediction_response_df: data frame containing for each ligand-setting combination the ligand importance scores for the individual importance scores, the complete model of importance scores and the ligand activity as well (TRUE or FALSE); $model: the caret model object that can be used on new importance scores to predict the ligand activity state.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
evaluation <- evaluate_importances_ligand_prediction(ligand_importances, "median", "lda")
print(head(evaluation))

## End(Not run)
```

evaluate_ligand_prediction_per_bin *Evaluate ligand activity predictions for different bins/groups of targets genes*

Description

evaluate_ligand_prediction_per_bin: Evaluate ligand activity predictions for different bins/groups of targets genes. Bins are constructed such that genes that are similarly frequently cited are grouped together and the different bins have similar size.

Usage

```
evaluate_ligand_prediction_per_bin(nbins,settings,ligand_target_matrix,ncitations,ligands_position =
```

Arguments

nbins	The number of bins the target genes should be divided in based on popularity.
settings	list of lists for which each sub-list contains the information about (expression) datasets; with minimally the following elements: name of the setting ($name), ligands (possibly) active in the setting of interest ($from).
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ncitations	A data frame denoting the number of times a gene is mentioned in the Pubmed literature. Should at least contain following variables: 'symbol' and 'ncitations'. Default: ncitations (variable contained in this package). See function get_ncitations_genes for a function that makes this data frame from current Pubmed information.
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
...	Additional arguments to make_discrete_ligand_target_matrix.

Value

A data.frame containing several classification evaluation metrics for ligand activity prediction. Predictions were evaluated for n different bins of target genes. The specific bin is indicated in the variable target_bin_id. target_bin_id = 1: target genes that are least mentioned in the Pubmed literature.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:10], convert_expression_settings_evaluation)
```

```
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
# ncitations = get_ncitations_genes()
ligand_prediction_performances_target_bins_popularity <- evaluate_ligand_prediction_per_bin(5, settings, ligand_

## End(Not run)
```

`evaluate_model` *Construct and evaluate a ligand-target model given input parameters.*

Description

`evaluate_model` will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input validation settings (both target gene prediction and ligand activity prediction).

Usage

```
evaluate_model(parameters_setting, lr_network, sig_network, gr_network, settings, calculate_popularit
```

Arguments

`parameters_setting`	A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See `prepare_settings_leave_o` and `add_hyperparameters_parameter_settings`.
`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)
`gr_network`	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
`settings`	A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
`calculate_popularity_bias_target_prediction`	Indicate whether popularity bias in target gene prediction performance should be calculated (TRUE or FALSE).
`calculate_popularity_bias_ligand_prediction`	Indicate whether popularity bias in ligand activity prediction performance should be calculated (TRUE or FALSE).

ncitations — A data frame denoting the number of times a gene is mentioned in the Pubmed literature. Should at least contain following variables: 'symbol' and 'ncitations'. Default: ncitations (variable contained in this package). See function get_ncitations_genes for a function that makes this data frame from current Pubmed information.

secondary_targets — Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

remove_direct_links — Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

n_target_bins — Indicate the number of bins the target genes will be divided in according to popularity. Only relevant when calculate_popularity_bias_target_prediction and/or calculate_popularity_bias_ligand_prediction is/are TRUE. Default = 3.

... — Additional arguments to make_discrete_ligand_target_matrix.

Value

A list containing following elements: $model_name, $performances_target_prediction, $performances_ligand_prediction, $performances_ligand_prediction_single

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network

## End(Not run)
```

---

evaluate_model_application

Construct and evaluate a ligand-target model given input parameters (for application purposes).

Description

evaluate_model_application will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input application settings (only target gene prediction).

Usage

```
evaluate_model_application(parameters_setting, lr_network, sig_network, gr_network, settings, seconda
```

Arguments

parameters_setting	A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See prepare_settings_leave_o and add_hyperparameters_parameter_settings.
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
settings	A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
secondary_targets	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE
remove_direct_links	Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"
...	Additional arguments to make_discrete_ligand_target_matrix.

Value

A list containing following elements: $model_name, $performances_target_prediction.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model_application, lr_network,

## End(Not run)
```

evaluate_model_application_multi_ligand *Construct and evaluate a ligand-target model given input parameters (for application purposes + multi-ligand predictive model).*

Description

evaluate_model_application_multi_ligand will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input application settings (only target gene prediction; multi-ligand classification).

Usage

```
evaluate_model_application_multi_ligand(parameters_setting, lr_network, sig_network, gr_network, sett
```

Arguments

parameters_setting	A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See prepare_settings_leave_o and add_hyperparameters_parameter_settings.
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
settings	A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
secondary_targets	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE
remove_direct_links	Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"
classification_algorithm	The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: with embedded feature selection: "rf","glm","fda","glmnet","sdwd","gam","glmboost", "pls" (load

"pls" package before!); without: "lda","naive_bayes", "pcaNNet". Please notice that not all these algorithms work when the features (i.e. ligand vectors) are categorical (i.e. discrete class assignments).

... Optional arguments to evaluate_multi_ligand_target_prediction.

Value

A list containing following elements: $model_name, $performances_target_prediction.

Examples

```
## Not run:
library(dplyr)
settings <- convert_expression_settings_evaluation(expression_settings_validation$TGFB_IL6_timeseries) %>% list(
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model_application_multi_ligan

## End(Not run)
```

---

evaluate_model_cv *Construct and evaluate a ligand-target model given input parameters with the purpose of evaluating cross-validation models.*

Description

evaluate_model_cv will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and calculate the model's performance in target gene prediction and feature importance scores for ligand prediction).

Usage

```
evaluate_model_cv(parameters_setting, lr_network, sig_network, gr_network, settings, secondary_target
```

Arguments

parameters_setting
: A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See prepare_settings_leave_o and add_hyperparameters_parameter_settings.

lr_network
: A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)

sig_network
: A data frame / tibble containing signaling interactions (required columns: from, to, source)

gr_network
: A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)

settings — A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.

secondary_targets — Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

remove_direct_links — Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

... — Additional arguments to make_discrete_ligand_target_matrix.

Value

A list containing following elements: $performances_target_prediction, $importances_ligand_prediction.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model_cv, lr_network, sig_netw

## End(Not run)
```

---

evaluate_multi_ligand_target_prediction

Evaluation of target gene prediction for multiple ligands.

Description

evaluate_multi_ligand_target_prediction Evaluate how well a trained model is able to predict the observed response to a combination of ligands (e.g. the set of DE genes after treatment of cells by multiple ligands). A classificiation algorithm chosen by the user is trained to construct one model based on the target gene predictions of all ligands of interest (ligands are considered as features). Several classification evaluation metrics for the prediction are calculated depending on whether the input ligand-target matrix contains probability scores for targets or discrete target assignments. In addition, variable importance scores can be extracted to rank the possible active ligands in order of importance for response prediction.

Usage

```
evaluate_multi_ligand_target_prediction(setting,ligand_target_matrix, ligands_position = "cols", algo
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
ligand_target_matrix	A matrix of ligand-target probabilty scores (recommended) or discrete target assignments (not-recommended).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
algorithm	The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: with embedded feature selection: "rf","glm","fda","glmnet","sdwd","gam","glmboost", "pls" (load "pls" package before!); without: "lda","naive_bayes", "pcaNNet". Please notice that not all these algorithms work when the features (i.e. ligand vectors) are categorical (i.e. discrete class assignments).
var_imps	Indicate whether in addition to classification evaluation performances, variable importances should be calculated. Default: TRUE.
cv	Indicate whether model training and hyperparameter optimization should be done via cross-validation. Default: TRUE. FALSE might be useful for applications only requiring variable importance, or when final model is not expected to be extremely overfit.
cv_number	The number of folds for the cross-validation scheme: Default: 4; only relevant when cv == TRUE.
cv_repeats	The number of repeats during cross-validation. Default: 2; only relevant when cv == TRUE.
parallel	Indiciate whether the model training will occur parallelized. Default: FALSE. TRUE only possible for non-windows OS.
n_cores	The number of cores used for parallelized model training via cross-validation. Default: 4. Only relevant on non-windows OS.
ignore_errors	Indiciate whether errors during model training by caret should be ignored such that another model training try will be initiated until model is trained without raising errors. Default: FALSE.
continuous	Indicate whether during training of the model, model training and evaluation should be done on class probabilities or discrete class labels. For huge class imbalance, we recommend setting this value to TRUE. Default: TRUE.

Value

A list with the following elements. $performances: data frame containing classification evaluation measure for classification on the test folds during training via cross-validation; $performances_training: data frame containing classification evaluation measures for classification of the

final model (discrete class assignments) on the complete data set (performance can be severly optimistic due to overfitting!); $performance_training_continuous: data frame containing classification evaluation measures for classification of the final model (class probability scores) on the complete data set (performance can be severly optimistic due to overfitting!) $var_imps: data frame containing the variable importances of the different ligands (embbed importance score for some classification algorithms, otherwise just the auroc); $prediction_response_df: data frame containing for each gene the ligand-target predictions of the individual ligands, the complete model and the response as well; $setting: name of the specific setting that needed to be evaluated; $ligands: ligands of interest.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
setting <- convert_expression_settings_evaluation(expression_settings_validation$TGFB_IL6_timeseries) %>% list()
ligands <- extract_ligands_from_settings(setting)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
output <- lapply(setting, evaluate_multi_ligand_target_prediction, ligand_target_matrix, ligands_position = "cols
output <- lapply(setting, evaluate_multi_ligand_target_prediction, make_discrete_ligand_target_matrix(ligand_tar

## End(Not run)
```

---

evaluate_multi_ligand_target_prediction_regression

Evaluation of target gene value prediction for multiple ligands (regression).

Description

`evaluate_multi_ligand_target_prediction_regression` Evaluate how well a trained model is able to predict the observed response to a combination of ligands (e.g. the absolute log fold change value of genes after treatment of cells by a ligand). A regression algorithm chosen by the user is trained to construct one model based on the target gene predictions of all ligands of interest (ligands are considered as features). It shows several regression model fit metrics for the prediction. In addition, variable importance scores can be extracted to rank the possible active ligands in order of importance for response prediction.

Usage

```
evaluate_multi_ligand_target_prediction_regression(setting,ligand_target_matrix, ligands_position =
```

Arguments

`setting`	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.

ligand_target_matrix
: A matrix of ligand-target probabilty scores (recommended) or discrete target assignments (not-recommended).

ligands_position
: Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

algorithm
: The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: "lm","glmnet", "rf".

var_imps
: Indicate whether in addition to classification evaluation performances, variable importances should be calculated. Default: TRUE.

cv
: Indicate whether model training and hyperparameter optimization should be done via cross-validation. Default: TRUE. FALSE might be useful for applications only requiring variable importance, or when final model is not expected to be extremely overfit.

cv_number
: The number of folds for the cross-validation scheme: Default: 4; only relevant when cv == TRUE.

cv_repeats
: The number of repeats during cross-validation. Default: 2; only relevant when cv == TRUE.

parallel
: Indiciate whether the model training will occur parallelized. Default: FALSE. TRUE only possible for non-windows OS.

n_cores
: The number of cores used for parallelized model training via cross-validation. Default: 4. Only relevant on non-windows OS.

ignore_errors
: Indiciate whether errors during model training by caret should be ignored such that another model training try will be initiated until model is trained without raising errors. Default: FALSE.

Value

A list with the following elements. $performances: data frame containing regression model fit metrics for regression on the test folds during training via cross-validation; $performances_training: data frame containing model fit metrics for regression of the final model on the complete data set (performance can be severly optimistic due to overfitting!); $var_imps: data frame containing the variable importances of the different ligands (embbed importance score for some classification algorithms, otherwise just the auroc); $prediction_response_df: data frame containing for each gene the ligand-target predictions of the individual ligands, the complete model and the response as well; $setting: name of the specific setting that needed to be evaluated; $ligands: ligands of interest.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
setting <- convert_expression_settings_evaluation_regression(expression_settings_validation$TGFB_IL6_timeseries
ligands <- extract_ligands_from_settings(setting)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
output <- lapply(setting, evaluate_multi_ligand_target_prediction_regression, ligand_target_matrix, ligands_posi

## End(Not run)
```

evaluate_random_model *Construct and evaluate a randomised ligand-target model given input parameters.*

Description

evaluate_random_model will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input validation settings (both target gene prediction and ligand activity prediction) after randomisation of the networks by edge swapping.

Usage

```
evaluate_random_model(parameters_setting, lr_network, sig_network, gr_network, settings, calculate_po
```

Arguments

parameters_setting	A list containing following elements: $model_name, $source_weights, $lr_sig_hub, $gr_hub, $ltf_cutoff, $algorithm, $damping_factor, $correct_topology. See prepare_settings_leave_o and add_hyperparameters_parameter_settings.
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
settings	A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
calculate_popularity_bias_target_prediction	Indicate whether popularity bias in target gene prediction performance should be calculated (TRUE or FALSE).
calculate_popularity_bias_ligand_prediction	Indicate whether popularity bias in ligand activity prediction performance should be calculated (TRUE or FALSE).
ncitations	A data frame denoting the number of times a gene is mentioned in the Pubmed literature. Should at least contain following variables: 'symbol' and 'ncitations'. Default: ncitations (variable contained in this package). See function get_ncitations_genes for a function that makes this data frame from current Pubmed information.
secondary_targets	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix

multiplication step considering primary targets as possible regulators). Default: FALSE

`remove_direct_links`
: Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

`n_target_bins`
: Indicate the number of bins the target genes will be divided in according to popularity. Only relevant when calculate_popularity_bias_target_prediction and/or calculate_popularity_bias_ligand_prediction is/are TRUE. Default = 3.

`...`
: Additional arguments to `make_discrete_ligand_target_matrix`.

Value

A list containing following elements: $model_name, $performances_target_prediction, $performances_ligand_prediction, $performances_ligand_prediction_single

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:4], convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_random_model, lr_network, sig_

## End(Not run)
```

`evaluate_single_importances_ligand_prediction`

Evaluation of ligand activity prediction performance of single ligand importance scores: aggregate all datasets.

Description

`evaluate_single_importances_ligand_prediction` Evaluate how well a single ligand importance score is able to predict the true activity state of a ligand. For this it is assumed, that ligand importance measures for truely active ligands will be higher than for non-active ligands. Several classification evaluation metrics for the prediction are calculated and variable importance scores can be extracted to rank the different importance measures in order of importance for ligand activity state prediction.

Usage

```
evaluate_single_importances_ligand_prediction(importances,normalization)
```

Arguments

importances	A data frame containing at least folowing variables: $setting, $test_ligand, $ligand and one or more feature importance scores. $test_ligand denotes the name of a possibly active ligand, $ligand the name of the truely active ligand.
normalization	Way of normalization of the importance measures: "mean" (classifcal z-score) or "median" (modified z-score) or "no" (use unnormalized feature importance scores - only recommended when evaluating ligand activity prediction on individual datasets)

Value

A data frame containing classification evaluation measures for the ligand activity state prediction single, individual feature importance measures.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
evaluation <- evaluate_single_importances_ligand_prediction(ligand_importances, normalization = "median")
print(head(evaluation))

## End(Not run)
```

---

evaluate_target_prediction

Evaluation of target gene prediction.

Description

evaluate_target_prediction Evaluate how well the model (i.e. the inferred ligand-target probability scores) is able to predict the observed response to a ligand (e.g. the set of DE genes after treatment of cells by a ligand). It shows several classification evaluation metrics for the prediction. Different classification metrics are calculated depending on whether the input ligand-target matrix contains probability scores for targets or discrete target assignments.

Usage

```
evaluate_target_prediction(setting,ligand_target_matrix, ligands_position = "cols")
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A data.frame with following variables: setting, ligand nd for probabilistic predictions: auroc, aupr, aupr_corrected (aupr - aupr for random prediction), sensitivity_roc (proxy measure, inferred from ROC), specificity_roc (proxy measure, inferred from ROC), mean_rank_GST_log_pval (-log10 of p-value of mean-rank gene set test), pearson (correlation coefficient), spearman (correlation coefficient); whereas for categorical predictions: accuracy, recall, specificity, precision, F1, F0.5, F2, mcc, informedness, markedness, fisher_pval_log (which is -log10 of p-value fisher exact test), fisher odds.
"mean_rank_GST_log_pval" will only be included in the dataframe if limma is installed. From NicheNet v2.1.7 onwards, limma is no longer a hard dependency of NicheNet.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
setting <- lapply(expression_settings_validation[1], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(setting)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
perf1 <- lapply(setting, evaluate_target_prediction, ligand_target_matrix)
print(head(perf1))
perf2 <- lapply(setting, evaluate_target_prediction, make_discrete_ligand_target_matrix(ligand_target_matrix))

## End(Not run)
```

---

evaluate_target_prediction_interprete

Evaluation of target gene prediction.

Description

evaluate_target_prediction_interprete Evaluate how well the model (i.e. the inferred ligand-target probability scores) is able to predict the observed response to a ligand (e.g. the set of DE genes after treatment of cells by a ligand; or the log fold change values). It shows several classification evaluation metrics for the prediction when response is categorical, or several regression model fit metrics when the response is continuous.

Usage

```
evaluate_target_prediction_interprete(setting,ligand_target_matrix, ligands_position = "cols")
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A list with the elements $performances and $prediction_response_df. $performance is a data.frame with classification evaluation metrics if response is categorical, or regression model fit metrics if response is continuous. $prediction_response_df shows for each gene, the model prediction and the response value of the gene (e.g. whether the gene the gene is a target or not according to the observed response, or the absolute value of the log fold change of a gene).

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
setting <- lapply(expression_settings_validation[1], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(setting)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
perf1 <- lapply(setting, evaluate_target_prediction_interprete, ligand_target_matrix)
setting <- lapply(expression_settings_validation[1], convert_expression_settings_evaluation_regression)
perf2 <- lapply(setting, evaluate_target_prediction_interprete, ligand_target_matrix)

## End(Not run)
```

---

evaluate_target_prediction_per_bin

Evaluate target gene predictions for different bins/groups of targets genes

---

Description

evaluate_target_prediction_per_bin: Evaluate target gene predictions for different bins/groups of targets genes. Bins are constructed such that genes that are similarly frequently cited are grouped together and the different bins have similar size.

Usage

```
evaluate_target_prediction_per_bin(nbins,settings,ligand_target_matrix,ncitations,ligands_position =
```

Arguments

nbins	The number of bins the target genes should be divided in based on popularity.
settings	list of lists for which each sub-list contains the information about (expression) datasets; with minimally the following elements: name of the setting ($name), ligands (possibly) active in the setting of interest ($from).
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ncitations	A data frame denoting the number of times a gene is mentioned in the Pubmed literature. Should at least contain following variables: 'symbol' and 'ncitations'. Default: ncitations (variable contained in this package). See function get_ncitations_genes for a function that makes this data frame from current Pubmed information.
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A data.frame containing several classification evaluation metrics for target gene prediction. Predictions were evaluated for n different bins of target genes. The specific bin is indicated in the variable target_bin_id. target_bin_id = 1: target genes that are least mentioned in the Pubmed literature.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:10], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
# ncitations = get_ncitations_genes()
performances_target_bins_popularity <- evaluate_target_prediction_per_bin(5, settings, ligand_target_matrix, nci

## End(Not run)
```

---

evaluate_target_prediction_regression

Evaluation of target gene value prediction (regression).

Description

evaluate_target_prediction_regression Evaluate how well the model (i.e. the inferred ligand-target probability scores) is able to predict the observed response to a ligand (e.g. the absolute log fold change value of genes after treatment of cells by a ligand). It shows several regression model fit metrics for the prediction.

Usage

```
evaluate_target_prediction_regression(setting,ligand_target_matrix, ligands_position = "cols")
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A data.frame with following variables: setting, ligand and as regression model fit metrics: r_squared: R squared, adj_r_squared: adjusted R squared, f_statistic: estimate of F-statistic, lm_coefficient_abs_t: absolute value of t-value of coefficient, inverse_rse: 1 divided by estimated standard deviation of the errors (inversed to become that higher values indicate better fit), reverse_aic: reverse value of Akaike information criterion (-AIC, to become that higher values indicate better fit), reverse_bic: the reverse value of the bayesian information criterion, inverse_mae: mean absolute error, pearson: pearson correlation coefficient, spearman: spearman correlation coefficient.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:2], convert_expression_settings_evaluation_regression)
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
perf1 <- lapply(settings, evaluate_target_prediction_regression, ligand_target_matrix)

## End(Not run)
```

expression_settings_validation

Expression datasets for validation

Description

A list

Usage

```
expression_settings_validation
```

Format

A list

diffexp **lfc** Log fold change (treated vs untreated)

qval Fdr corrected p-value

gene Gene symbol

from Gene symbol(s) of ligand(s) by which the cells were treated

name Name of the expression validation dataset

type Type of dataset based on response time: "primary", "secondary", "primary + secondary"

extract_ligands_from_settings

Extract ligands of interest from settings

Description

`extract_ligands_from_settings` Extract ligands of interest from (expression) settings in correct to construct the ligand-target matrix.

Usage

```
extract_ligands_from_settings(settings,combination = TRUE)
```

Arguments

`settings`	A list of lists for which each sub-list contains the information about (expression) datasets; with minimally the following elements: name of the setting ($name), ligands (possibly) active in the setting of interest ($from).
`combination`	Indicate whether in case multiple ligands are possibly active ligand combinations should be extracted or only individual ligands. Default: TRUE.

Value

A list containing the ligands and ligands combinations for which a ligand-target matrix should be constructed. When for a particular dataset multiple ligands are possibly active (i.e. more than ligand in .$from slot of sublist of settings), then both the combination of these multiple ligands and each of these multiple ligands individually will be select for model construction.

Examples

```
## Not run:
ligands <- extract_ligands_from_settings(expression_settings_validation)

## End(Not run)
```

---

extract_top_fraction_ligands

Get the predicted top n percentage ligands of a target of interest

---

Description

extract_top_fraction_ligands Get the predicted top n percentage ligands of a target of interest

Usage

```
extract_top_fraction_ligands(target_oi,top_fraction,ligand_target_matrix,ligands_position = "cols")
```

Arguments

target_oi	The target gene of interest of which top upstream ligands should be returned
top_fraction	A number between 0 and 1 indicating which top fraction of ligands should be returned.
ligand_target_matrix	A matrix of ligand-target probabilty scores.
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A named numeric vector of ligand-target gene probability scores of the top ligands.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
targets <- extract_top_fraction_ligands("ID3", 0.01, ligand_target_matrix)

## End(Not run)
```

---

extract_top_fraction_targets

Get the predicted top n percentage target genes of a ligand of interest

---

Description

extract_top_fraction_targets Get the predicted top n percentage target genes of a ligand of interest.

Usage

```
extract_top_fraction_targets(ligand_oi,top_fraction,ligand_target_matrix,ligands_position = "cols")
```

Arguments

ligand_oi	The ligand of interest of which top target genes should be returned
top_fraction	A number between 0 and 1 indicating which top fraction of target genes should be returned.
ligand_target_matrix	A matrix of ligand-target probabilty scores.
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A named numeric vector of ligand-target gene probability scores of the top target genes.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
targets <- extract_top_fraction_targets("BMP2", 0.01, ligand_target_matrix)

## End(Not run)
```

extract_top_n_ligands *Get the predicted top n ligands of a target gene of interest*

Description

extract_top_n_ligands Get the predicted top n ligands of a target gene of interest.

Usage

```
extract_top_n_ligands(target_oi,top_n,ligand_target_matrix,ligands_position = "cols")
```

Arguments

target_oi	The target gene of interest of which top upstream ligands should be returned
top_n	A number between 0 and the total nr of ligands indicating which top n of ligands should be returned.
ligand_target_matrix	A matrix of ligand-target probabilty scores.
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A named numeric vector of ligand-target gene probability scores of the top ligands.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
targets <- extract_top_n_ligands("BMP2", 2, ligand_target_matrix)

## End(Not run)
```

extract_top_n_targets *Get the predicted top n target genes of a ligand of interest*

Description

extract_top_n_targets Get the predicted top n target genes of a ligand of interest.

Usage

```
extract_top_n_targets(ligand_oi,top_n,ligand_target_matrix,ligands_position = "cols")
```

Arguments

ligand_oi	The ligand of interest of which top target genes should be returned
top_n	A number between 0 and the total nr of target genes indicating which top n of target genes should be returned.
ligand_target_matrix	A matrix of ligand-target probabilty scores.
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A named numeric vector of ligand-target gene probability scores of the top target genes.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
targets <- extract_top_n_targets("BMP2", 50, ligand_target_matrix)

## End(Not run)
```

---

geneinfo_2022 *Gene annotation information: version 2 - january 2022*

Description

A data.frame/tibble describing HGNC human gene symbols, their entrez ids and the MGI mouse gene symbols and entrez ids of the one-one orthologs as determined via NCBI's homologene db and biomaRt ensembl db.

Usage

```
geneinfo_2022
```

Format

A data frame/tibble

symbol human gene symbol

entrez human gene entrez

entrez_mouse mouse homolog gene entrez

symbol_mouse mouse homolog gene symbol

geneinfo_alias_human — *Gene annotation information: version 2 - january 2022 - suited for alias conversion*

Description

A data.frame/tibble describing HGNC human gene symbols, their entrez ids and potential aliases.

Usage

```
geneinfo_alias_human
```

Format

A data frame/tibble

symbol human gene symbol

entrez human gene entrez

alias human gene alias

geneinfo_alias_mouse — *Gene annotation information: version 2 - january 2022 - suited for alias conversion*

Description

A data.frame/tibble describing MGI mouse gene symbols, their entrez ids and potential aliases.

Usage

```
geneinfo_alias_mouse
```

Format

A data frame/tibble

symbol mouse gene symbol

entrez mouse gene entrez

alias mouse gene alias

geneinfo_human *Gene annotation information*

Description

A data.frame/tibble describing HGNC human gene symbols, their entrez ids and the MGI mouse gene symbols and entrez ids of the one-one orthologs as determined via NCBI's homologene db and biomaRt ensembl db.

Usage

```
geneinfo_human
```

Format

A data frame/tibble

symbol human gene symbol

entrez human gene entrez

entrez_mouse mouse homolog gene entrez

symbol_mouse mouse homolog gene symbol

generate_info_tables *Generate tables used for* generate_prioritization_tables

Description

Calculate differential expression, average expression, and condition specificity of ligands and receptors.

Usage

```
generate_info_tables(
  seuratObj,
  celltype_colname,
  senders_oi,
  receivers_oi,
  lr_network_filtered,
  condition_colname = NULL,
  condition_oi = NULL,
  condition_reference = NULL,
  scenario = "case_control",
  assay_oi = NULL,
  ...
)
```

Arguments

seuratObj	Seurat object
celltype_colname	Name of the meta data column that indicates the cell type of a cell
senders_oi	Default NULL: all celltypes will be considered as senders. If you want to select specific senders of interest: you can add this here as character vector.
receivers_oi	Default NULL: all celltypes will be considered as receivers If you want to select specific receivers of interest: you can add this here as character vector.
lr_network_filtered	Ligand-receptor network that has been filtered to only contain ligands and receptors that are expressed
condition_colname	Name of the meta data column that indicates from which group/condition a cell comes from
condition_oi	If provided, subset seurat_obj so DE is only calculated for cells belonging to condition_oi
condition_reference	Reference condition for condition specificity calculation
scenario	"case_control" or "one_condition": if "case_control", calculate condition specificity. If "one_condition", only calculate cell type specificity.
assay_oi	Which assay need to be used for DE calculation. If NULL, will use DefaultAssay
...	Arguments passed to FindAllMarkers, FindMarkers, and AverageExpression

Value

List of dataframes containing sender-receiver DE, sender-receiver expression, and condition DE

---

generate_prioritization_tables

Perform a prioritization of cell-cell interactions (similar to MultiNicheNet).

Description

User can choose the importance attached to each of the following prioritization criteria: differential expression of ligand and receptor, cell-type specificity of expression of ligand and receptor, NicheNet ligand activity

Usage

```
generate_prioritization_tables(
  sender_receiver_info,
  sender_receiver_de,
  ligand_activities,
  lr_condition_de = NULL,
  prioritizing_weights = NULL,
  scenario = "case_control"
)
```

Arguments

sender_receiver_info	Output of generate_info_tables OR get_exprs_avg -> process_table_to_ic
sender_receiver_de	Output of generate_info_tables OR calculate_de -> process_table_to_ic
ligand_activities	Output of predict_ligand_activities
lr_condition_de	Output of generate_info_tables OR FindMarkers -> process_table_to_ic
prioritizing_weights	Named vector indicating the relative weights of each prioritization criterion (default: NULL). If NULL, the weights are determined by the chosen "scenario". If provided, the vector must contain the following names: "de_ligand", "de_receptor", "activity_scaled", "exprs_ligand", "exprs_receptor", "ligand_condition_specificity", "receptor_condition_specificity"
scenario	"case_control" or "one_condition": if "case_control", all weights are set to 1. If "one_condition", the weights are set to 0 for condition specificity and 1 for the remaining criteria.

Value

Data frames of prioritized sender-ligand-receiver-receptor interactions. The resulting dataframe contains columns from the input dataframes, but columns from lr_condition_de are suffixed with _group (some columns from lr_condition_de are also not present). Additionally, the following columns are added:

- lfc_pval_*: product of -log10(pval) and the LFC of the ligand/receptor
- p_val_adapted_*: p-value adapted to the sign of the LFC to only consider interactions where the ligand/receptor is upregulated in the sender/receiver
- activity_zscore: z-score of the ligand activity
- prioritization_score: The prioritization score for each interaction, calculated as a weighted sum of the prioritization criteria.

Moreover, scaled_* columns are scaled using the corresponding column's ranking or the scale_quantile_adapted function. The columns used for prioritization are scaled_p_val_adapted_ligand, scaled_p_val_adapted_receptor, scaled_activity, scaled_avg_exprs_ligand, scaled_avg_exprs_receptor, scaled_p_val_adapted_ligand_group, scaled_p_val_adapted_receptor_group

Examples

```
## Not run:

# Calculate tables with generate_info_tables
info_tables <- generate_info_tables(seurat_obj, "celltype", sender_celltypes, receiver, expressed_ligands, expres

# Generate prioritization tables
generate_prioritization_tables(info_tables$sender_receiver_info,
  info_tables$sender_receiver_de,
  ligand_activities,
  info_tables$lr_condition_de,
  scenario = "case_control"
)

# Alternatively, these tables can also be calculated manually:
# Calculate LCMV-specific average expression
expression_info <- get_exprs_avg(seurat_obj, "celltype", condition_oi = "LCMV", condition_colname = "aggregate")

# Calculate LCMV-specific cell-type markers
DE_table <- calculate_de(seurat_obj, "celltype", condition_oi = "LCMV", condition_colname = "aggregate")

# Calculate condition-specific markers
condition_markers <- FindMarkers(
  object = seuratObj, ident.1 = "LCMV", ident.2 = "SS",
  group.by = "aggregate", min.pct = 0, logfc.threshold = 0
) %>% rownames_to_column("gene")

# Process tables
processed_expr_info <- process_table_to_ic(expression_info, table_type = "expression", lr_network)
processed_DE_table <- process_table_to_ic(DE_table,
  table_type = "celltype_DE", lr_network,
  senders_oi = sender_celltypes, receivers_oi = receiver
)
processed_condition_DE_table <- process_table_to_ic(condition_markers, table_type = "group_DE", lr_network)

# Custom weights
prioritizing_weights <- c("de_ligand" = 1, "de_receptor" = 1, "activity_scaled" = 2, "exprs_ligand" = 1, "exprs_rec

generate_prioritization_tables(processed_expr_info, processed_DE_table, ligand_activities, processed_condition_

## End(Not run)
```

---

get_active_ligand_receptor_network

Get active ligand-receptor network for cellular interaction between a sender and receiver cell.

Description

`get_active_ligand_receptor_network` Get active ligand-receptor network by looking for which ligands are expressed in a sender/signaling cell and which receptors are expressed in the receiver

cell. Instead of looking at absolute expression, it is possible as well to extract a ligand-receptor network of differentially expressed ligands and receptors if a vector of log2 fold change values is used as input.

Usage

```
get_active_ligand_receptor_network(expression_sender, expression_receiver, lr_network, expression_cu
```

Arguments

expression_sender	A named numeric vector of gene expression levels (absolute or logfc) for the signaling cell that sends extracellular signals to the receiver cell
expression_receiver	A named numeric vector of gene expression levels (absolute or logfc) for the receiver cell that receives extracellular signals from the sender cell
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to). Can be both unweighted and weighted.
expression_cutoff_sender	The cutoff on expression value for the sender cell: ligands will be considered active if their expression is higher than the cutoff. Default: 0.
expression_cutoff_receiver	The cutoff on expression value for the receiver cell: receptors will be considered active if their expression is higher than the cutoff. Default: 0.

Value

A data frame containing at least the variables from, to, sender_expression, receiver_expression. In this network, the active ligand-receptor interactions in the system of interest are shown.

Examples

```
## Not run:
library(dplyr)
expression_vector_sender <- rnorm(n = 10000, mean = 6, sd = 3)
expression_vector_receiver <- rnorm(n = 10000, mean = 6, sd = 3)
names(expression_vector_sender) <- sample(x = geneinfo_human$symbol, size = 10000, replace = FALSE)
names(expression_vector_receiver) <- sample(x = geneinfo_human$symbol, size = 10000, replace = FALSE)
weighted_lr_network <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df, n_out
sender_cell_receiver_lr_network <- get_active_ligand_receptor_network(expression_vector_sender, expression_vect

## End(Not run)
```

get_active_ligand_target_df *Get active ligand-target network in data frame format.*

Description

get_active_ligand_target_df Get active ligand-target network, meaning that only target genes that are part of the response of interest will be kept as target genes. In addition, target genes with probability scores beneath a predefined cutoff will be removed from the nework.

Usage

```
get_active_ligand_target_df(response,ligand_target_matrix, ligands_position = "cols", cutoff = 0)
```

Arguments

response	A named logical vector indicating whether a gene is responding in a biological system or not (e.g. DE after cell-cell interaction)
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
cutoff	A number indicating how high ligand-target probability scores should be in order to be kept. Default: 0. When 0 is used as cutoff and the input matrix is a discrete ligand-target assingment matrix, only TRUE interactions will be kept.

Value

A data frame representing the active ligand-target network; with variables $ligand, $target and $score.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
setting <- lapply(expression_settings_validation[1:2], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(setting)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_target_matrix_discrete <- make_discrete_ligand_target_matrix(ligand_target_matrix)
active_lt_df <- get_active_ligand_target_df(setting[[1]] %>% .$response, ligand_target_matrix_discrete)

## End(Not run)
```

get_active_ligand_target_matrix *Get active ligand-target matrix.*

Description

get_active_ligand_target_matrix Get active ligand-target matrix, meaning that only target genes part of the response of interest will be kept as target genes in the input ligand-target matrix.

Usage

```
get_active_ligand_target_matrix(response,ligand_target_matrix, ligands_position = "cols")
```

Arguments

response	A named logical vector indicating whether a gene is responding in a biological system or not (e.g. DE after cell-cell interaction)
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A matrix with ligand-target probability scores (or discrete ligand-target assignments) for the active target genes in the system of interest.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
setting <- lapply(expression_settings_validation[1:2], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(setting)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
active_lt <- get_active_ligand_target_matrix(setting[[1]] %>% .$response, ligand_target_matrix)

## End(Not run)
```

get_active_regulatory_network *Get active gene regulatory network in a receiver cell.*

Description

get_active_regulatory_network Get active regulatory network by looking for which TFs/regulatory genes are expressed in the receiver cell. Instead of looking at absolute expression, it is possible as well to extract a gene regulatory network of differentially expressed TFs if a vector of log2 fold change values is used as input.

Usage

```
get_active_regulatory_network(expression_receiver, gr_network, expression_cutoff_receiver = 0)
```

Arguments

expression_receiver	A named numeric vector of gene expression levels (absolute or logfc) for the receiver cell that receives extracellular signals from the sender cell
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to). Can be both unweighted and weighted.
expression_cutoff_receiver	The cutoff on expression value for the receiver cell: receptors will be considered active if their expression is higher than the cutoff. Default: 0.

Value

A data frame containing at least the variables from, to, receiver_expression. In this network, the active gene regulatory interactions in the system of interest are shown.

Examples

```
## Not run:
library(dplyr)
expression_vector_receiver <- rnorm(n = 10000, mean = 6, sd = 3)
names(expression_vector_receiver) <- sample(x = geneinfo_human$symbol, size = 10000, replace = FALSE)
receiver_gr_network <- get_active_regulatory_network(expression_vector_receiver, gr_network, expression_cutoff_r

## End(Not run)
```

get_active_signaling_network *Get active signaling network in a receiver cell.*

Description

get_active_signaling_network Get active signaling network by looking for which signaling-related genes are expressed in the receiver cell. Instead of looking at absolute expression, it is possible as well to extract a signaling network of differentially expressed signaling mediators if a vector of log2 fold change values is used as input.

Usage

```
get_active_signaling_network(expression_receiver, sig_network, expression_cutoff_receiver = 0)
```

Arguments

expression_receiver	A named numeric vector of gene expression levels (absolute or logfc) for the receiver cell that receives extracellular signals from the sender cell
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to). Can be both unweighted and weighted.
expression_cutoff_receiver	The cutoff on expression value for the receiver cell: receptors will be considered active if their expression is higher than the cutoff. Default: 0.

Value

A data frame containing at least the variables from, to, receiver_expression. In this network, the active signaling interactions in the system of interest are shown.

Examples

```
## Not run:
library(dplyr)
expression_vector_receiver <- rnorm(n = 10000, mean = 6, sd = 3)
names(expression_vector_receiver) <- sample(x = geneinfo_human$symbol, size = 10000, replace = FALSE)
receiver_sig_network <- get_active_signaling_network(expression_vector_receiver, sig_network, expression_cutoff_

## End(Not run)
```

get_database_cache_stats *Get Cache Statistics*

Description

Shows information about cached databases.

Usage

```
get_database_cache_stats()
```

get_expressed_genes *Determine expressed genes of a cell type from an input object*

Description

Return the genes that are expressed in given cell cluster(s) based on the fraction of cells in the cluster(s) that should express the cell.

Usage

```
get_expressed_genes(celltype_oi, object, ...)

## Default S3 method:
get_expressed_genes(celltype_oi, object, celltype_annot, pct = 0.1)

## S3 method for class 'Seurat'
get_expressed_genes(celltype_oi, seurat_obj, pct = 0.1, assay_oi = NULL, ...)
```

Arguments

celltype_oi	Character vector of cell type(s) to be considered. If input is a Seurat object, these must correspond to the cell identities from Idents.
object	Input matrix with rows as genes and columns as cells
...	additional parameters passed to GetAssayData (in case the slot/layer needs to be specified)
celltype_annot	Vector of cell type annotations
pct	We consider genes expressed if they are expressed in at least a specific fraction of cells of the given cluster(s). This number indicates this fraction. Default: 0.10. Choice of this parameter is important and depends largely on the used sequencing platform. We recommend to require a lower fraction (like the default 0.10) for 10X data than for e.g. Smart-seq2 data.
seurat_obj	Single-cell expression or spatial dataset as Seurat object
assay_oi	If wanted: specify yourself which assay to look for. If not NULL, the DefaultAssay of the Seurat object is used.

Value

A vector containing gene symbols of the expressed genes

Examples

```
## Not run:
# For sparse matrix
get_expressed_genes("CD8 T", GetAssayData(seuratObj), seuratObj$celltype, pct = 0.10)

## End(Not run)

## Not run:
# For Seurat object
get_expressed_genes(celltype_oi = "CD8 T", seurat_obj = seuratObj, pct = 0.10)

## End(Not run)
```

---

get_exprs_avg *Calculate average of gene expression per cell type.*

Description

get_exprs_avg Calculate average of gene expression per cell type. If condition_oi is provided, only consider cells from that condition.

Usage

```
get_exprs_avg(
  seurat_obj,
  celltype_colname,
  condition_oi = NULL,
  condition_colname = NULL,
  assay_oi = NULL,
  ...
)
```

Arguments

seurat_obj	Seurat object
celltype_colname	Name of the meta data column that indicates the cell type of a cell
condition_oi	If provided, subset seurat_obj so average expression is only calculated for cells belonging to condition_oi
condition_colname	Name of the meta data column that indicates from which group/condition a cell comes from

assay_oi	Which assay need to be used for DE calculation. If NULL, will use DefaultAssay
...	Arguments passed to Seurat::AverageExpression, usually for slot/layer to use (default: data)

Value

Data frame with average gene expression per cell type.

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj.rds"))
seurat_obj$celltype <- make.names(seuratObj$celltype)
# Calculate average expression across conditions
expression_info <- get_exprs_avg(seurat_obj, "celltype")
# Calculate LCMV-specific average expression
expression_info <- get_exprs_avg(seurat_obj, "celltype", condition_oi = "LCMV", condition_colname = "aggregate")

## End(Not run)
```

---

get_lfc_celltype *Get log fold change values of genes in cell type of interest*

Description

get_lfc_celltype Get log fold change of genes between two conditions in cell type of interest when using a Seurat single-cell object.

Usage

```
get_lfc_celltype(celltype_oi, seurat_obj, condition_colname, condition_oi, condition_reference, cellt
#'
```

Arguments

celltype_oi	Name of celltype of interest. Should be present in the celltype metadata dataframe.
seurat_obj	Single-cell expression dataset as Seurat object https://satijalab.org/seurat/.
condition_colname	Name of the column in the meta data dataframe that indicates which condition/sample cells were coming from.
condition_oi	Condition of interest. Should be a name present in the "condition_colname" column of the metadata.
condition_reference	The second condition (e.g. reference or steady-state condition). Should be a name present in the "condition_colname" column of the metadata.

celltype_col	Metadata colum name where the cell type identifier is stored. Default: "celltype". If this is NULL, the Idents() of the seurat object will be considered as your cell type identifier.
...	Additional arguments passed to FindMarkers.

Value

A tbl with the log fold change values of genes. Positive lfc values: higher in condition_oi compared to condition_reference.

Examples

```
## Not run:
requireNamespace("dplyr")
seuratObj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj_test.rds"))
get_lfc_celltype(seurat_obj = seuratObj, celltype_oi = "CD8 T", condition_colname = "aggregate", condition_oi = "L

## End(Not run)
```

---

get_ligand_activities_targets

Calculate the ligand activities and infer ligand-target links based on a list of niche-specific genes per receiver cell type

---

Description

get_ligand_activities_targets Calculate the ligand activities and infer ligand-target links based on a list of niche-specific genes per receiver cell type.

Usage

```
get_ligand_activities_targets(niche_geneset_list, ligand_target_matrix, top_n_target)
```

Arguments

niche_geneset_list	List of lists/niches giving the geneset of interest for the receiver cell type in each niche.
ligand_target_matrix	The NicheNet ligand-target matrix of the organism of interest denoting regulatory potential scores between ligands and targets (ligands in columns).
top_n_target	To predict active, affected targets of the prioritized ligands, consider only DE genes if they also belong to the a priori top n ("top_n_targets") targets of a ligand. Default = 200.

Value

A tibble of ligands, their activities and targets in each receiver cell type

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
DE_receiver <- calculate_niche_de(seurat_obj, niches, "receiver")
expression_pct <- 0.10
lfc_cutoff <- 0.15 # recommended for 10x as min_lfc cutoff.
specificity_score_targets <- "min_lfc"
DE_receiver_processed_targets <- process_receiver_target_de(DE_receiver_targets = DE_receiver, niches = niches, e
background <- DE_receiver_processed_targets %>%
  pull(target) %>%
  unique()
geneset_KC <- DE_receiver_processed_targets %>%
 filter(receiver == niches$KC_niche$receiver & target_score >= lfc_cutoff & target_significant == 1 & target_prese
  pull(target) %>%
  unique()
geneset_MoMac2 <- DE_receiver_processed_targets %>%
 filter(receiver == niches$MoMac2_niche$receiver & target_score >= lfc_cutoff & target_significant == 1 & target_p
  pull(target) %>%
  unique()
geneset_MoMac1 <- DE_receiver_processed_targets %>%
 filter(receiver == niches$MoMac1_niche$receiver & target_score >= lfc_cutoff & target_significant == 1 & target_p
  pull(target) %>%
  unique()
top_n_target <- 250
niche_geneset_list <- list(
  "KC_niche" = list(
    "receiver" = "KCs",
    "geneset" = geneset_KC,
    "background" = background
  ),
  "MoMac1_niche" = list(
    "receiver" = "MoMac1",
    "geneset" = geneset_MoMac1,
    "background" = background
  ),
  "MoMac2_niche" = list(
    "receiver" = "MoMac2",
    "geneset" = geneset_MoMac2,
```

```
    "background" = background
  )
)
ligand_activities_targets <- get_ligand_activities_targets(niche_geneset_list = niche_geneset_list, ligand_targe

## End(Not run)
```

`get_ligand_signaling_path` *Get ligand-target signaling paths between ligand(s) and target gene(s) of interest*

Description

`get_ligand_signaling_path` Extract possible signaling paths between a ligand and target gene of interest. The most highly weighted path(s) will be extracted.

Usage

```
get_ligand_signaling_path(ligand_tf_matrix, ligands_all, targets_all, top_n_regulators = 4, weighted_
```

Arguments

`ligand_tf_matrix`	A matrix of ligand-regulator probability scores
`ligands_all`	A character vector of one or more ligands of interest
`targets_all`	A character vector of one or more target genes of interest
`top_n_regulators`	The number of top regulators that should be included in the ligand-target signaling network. Top regulators are regulators that score both high for being upstream of the target gene(s) and high for being downstream of the ligand. Default: 4.
`weighted_networks`	A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
`ligands_position`	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols".
`minmax_scaling`	Indicate whether the weights of both dataframes should be min-max scaled between 0.75 and 1. Default: FALSE.

Value

A list containing 2 elements (sig and gr): the integrated weighted ligand-signaling and gene regulatory networks data frame / tibble format with columns: from, to, weight

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_tf_matrix <- construct_ligand_tf_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorit
all_ligands <- c("BMP2")
all_targets <- c("HEY1")
top_n_regulators <- 2
ligand_target_signaling_list <- get_ligand_signaling_path(ligand_tf_matrix, all_ligands, all_targets, top_n_regu

## End(Not run)
```

---

get_ligand_signaling_path_with_receptor

Get ligand-target signaling paths between ligand(s), receptors, and target gene(s) of interest

---

Description

`get_ligand_signaling_path_with_receptor` Extract possible signaling paths between a ligand(s), receptor(s) and target gene(s) of interest. The most highly weighted path(s) will be extracted.

Usage

```
get_ligand_signaling_path_with_receptor(ligand_tf_matrix, ligands_all, receptors_all, targets_all, to
```

Arguments

`ligand_tf_matrix`	A matrix of ligand-regulator probability scores
`ligands_all`	A character vector of one or more ligands of interest
`receptors_all`	A character vector of one or more receptors of interest
`targets_all`	A character vector of one or more target genes of interest
`top_n_regulators`	The number of top regulators that should be included in the ligand-target signaling network. Top regulators are regulators that score both high for being upstream of the target gene(s) and high for being downstream of the ligand. Default: 4.
`weighted_networks`	A list of two elements: lr_sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
`ligands_position`	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols".

Value

A list containing 2 elements (sig and gr): the integrated weighted ligand-signaling and gene regulatory networks data frame / tibble format with columns: from, to, weight

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_tf_matrix <- construct_ligand_tf_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorit
all_ligands <- c("BMP2")
all_receptors <- c("BMPR2")
all_targets <- c("HEY1")
top_n_regulators <- 2
ligand_target_signaling_list <- get_ligand_signaling_path_with_receptor(ligand_tf_matrix, all_ligands, all_recep

## End(Not run)
```

---

get_ligand_slope_ligand_prediction_popularity

Regression analysis between popularity of left-out ligands for ligand activity prediction performance

---

Description

`get_ligand_slope_ligand_prediction_popularity`: Performs regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of left-out ligands for ligand activity prediction performance.

Usage

```
get_ligand_slope_ligand_prediction_popularity(metric,performances)
```

Arguments

`metric`	The name of the performance metric of which the trend with the popularity of the target genes should be calculated.
`performances`	A data.frame in which the performance measures for target gene predictions of ligands are denoted together with the "popularity index" indicating which percentage of most popular ligands were left out.)

Value

A data.frame in which the regression coefficient estimate, p-value and corresponding R-squared value are shown for the regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of the left out ligands.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target

ligand_activity_popularity_bias <- lapply(0:3, ligand_activity_performance_top_i_removed, ligand_importances, nc
slopes_auroc <- get_ligand_slope_ligand_prediction_popularity("auroc", ligand_activity_popularity_bias)
slopes_df <- ligand_activity_popularity_bias %>%
  select(-importance_measure, -popularity_index) %>%
  colnames() %>%
 lapply(., get_ligand_slope_ligand_prediction_popularity, ligand_activity_popularity_bias) %>%
  bind_rows()

## End(Not run)
```

---

get_ligand_target_links_oi

Get ligand-target links of interest

Description

Filter ligand-target links based on a cutoff

Usage

```
get_ligand_target_links_oi(ligand_type_indication_df, active_ligand_target_links_df, cutoff = 0.40)
```

Arguments

ligand_type_indication_df	Dataframe with column names ligand_type and ligand, from the function assign_ligands_to_celltype
active_ligand_target_links_df	Dataframe with weighted ligand-target links from the function get_ligand_target_links, and an additional column target_type that indicates the grouping of target genes
cutoff	Quantile to filter ligand-target links (default = 0.40, meaning 40% of the lowest weighted ligand-target links are removed)

Value

A dataframe with ligand-target links with weights above a certain cutoff. This dataframe also contains the attribute `cutoff_include_all_ligands`, which is the cutoff value of regulatory potential used at `cutoff` quantile.

Examples

```
## Not run:
active_ligand_target_links_df <- lapply(best_upstream_ligands, get_weighted_ligand_target_links,
  geneset = geneset_oi,
  ligand_target_matrix = ligand_target_matrix,
  n = 200
)
active_ligand_target_links_df <- drop_na(bind_rows(active_ligand_target_links_df))
ligand_type_indication_df <- assign_ligands_to_celltype(seuratObj = seuratObj, ligands = best_upstream_ligands[1:
circos_links <- get_ligand_target_links_oi(ligand_type_indication_df,
  active_ligand_target_links_df %>% mutate(target_type = "LCMV-DE"),
  cutoff = 0.40
)
attr(circos_links, "cutoff_include_all_ligands") # This is the cutoff value of regulatory potential used

## End(Not run)
```

---

get_multi_ligand_importances

Get ligand importances from a multi-ligand classfication model.

Description

`get_multi_ligand_importances` A classificiation algorithm chosen by the user is trained to construct one model based on the target gene predictions of all ligands of interest (ligands are considered as features) in order to predict the observed response in a particular dataset. Variable importance scores that indicate for each ligand the importance for response prediction, are extracted. It can be assumed that ligands with higher variable importance scores are more likely to be a true active ligand.

Usage

```
get_multi_ligand_importances(setting,ligand_target_matrix, ligands_position = "cols", algorithm, cv =
```

Arguments

`setting`	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) of which the predictve performance need to be assessed; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest. $ligand: NULL or the name of the ligand(s) that are known to be active in the setting of interest.

ligand_target_matrix
: A matrix of ligand-target probabilty scores (recommended) or discrete target assignments (not-recommended).

ligands_position
: Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

algorithm
: The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: with embedded feature selection: "rf","glm","fda","glmnet","sdwd","gam","glmboost"; without: "lda","naive_bayes","pls"(b
bug in current version of pls package), "pcaNNet". Please notice that not all these algorithms work when the features (i.e. ligand vectors) are categorical (i.e. discrete class assignments).

cv
: Indicate whether model training and hyperparameter optimization should be done via cross-validation. Default: TRUE. FALSE might be useful for applications only requiring variable importance, or when final model is not expected to be extremely overfit.

cv_number
: The number of folds for the cross-validation scheme: Default: 4; only relevant when cv == TRUE.

cv_repeats
: The number of repeats during cross-validation. Default: 2; only relevant when cv == TRUE.

parallel
: Indiciate whether the model training will occur parallelized. Default: FALSE. TRUE only possible for non-windows OS.

n_cores
: The number of cores used for parallelized model training via cross-validation. Default: 4. Only relevant on non-windows OS.

ignore_errors
: Indiciate whether errors during model training by caret should be ignored such that another model training try will be initiated until model is trained without raising errors. Default: FALSE.

continuous
: Indicate whether during training of the model, model training and evaluation should be done on class probabilities or discrete class labels. For huge class imbalance, we recommend setting this value to TRUE. Default: TRUE.

known
: Indicate whether the true active ligand for a particular dataset is known or not. Default: TRUE. The true ligand will be extracted from the $ligand slot of the setting.

filter_genes
: Indicate whether 50 per cent of the genes that are the least variable in ligand-target scores should be removed in order to reduce the training of the model. Default: FALSE.

Value

A data.frame with for each ligand - data set combination, feature importance scores indicating how important the query ligand is for the prediction of the response in the particular dataset, when prediction is done via a trained classification model with all possible ligands as input. In addition to the importance score(s), the name of the particular setting ($setting), the name of the query ligand($test_ligand), the name of the true active ligand (if known: $ligand).

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances_glm <- dplyr::bind_rows(lapply(settings_ligand_pred, get_multi_ligand_importances, ligand_tar
print(head(ligand_importances_glm))

## End(Not run)
```

---

get_multi_ligand_importances_regression

Get ligand importances from a multi-ligand regression model.

Description

`get_multi_ligand_importances_regression` A regression algorithm chosen by the user is trained to construct one model based on the target gene predictions of all ligands of interest (ligands are considered as features) in order to predict the observed response in a particular dataset (respone: e.g. absolute value of log fold change). Variable importance scores that indicate for each ligand the importance for response prediction, are extracted. It can be assumed that ligands with higher variable importance scores are more likely to be a true active ligand.

Usage

```
get_multi_ligand_importances_regression(setting,ligand_target_matrix, ligands_position = "cols", algo
```

Arguments

`setting`	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) of which the predictve performance need to be assessed; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest. $ligand: NULL or the name of the ligand(s) that are known to be active in the setting of interest.
`ligand_target_matrix`	A matrix of ligand-target probabilty scores (recommended) or discrete target assignments (not-recommended).
`ligands_position`	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
`algorithm`	The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: with embedded feature selection: "rf","glm","fda","glmnet","sdwd","gam","glmboost"; without: "lda","naive_bayes","pls"(b bug in current version of pls package), "pcaNNet". Please notice that not all

these algorithms work when the features (i.e. ligand vectors) are categorical (i.e. discrete class assignments).

`cv` Indicate whether model training and hyperparameter optimization should be done via cross-validation. Default: TRUE. FALSE might be useful for applications only requiring variable importance, or when final model is not expected to be extremely overfit.

`cv_number` The number of folds for the cross-validation scheme: Default: 4; only relevant when cv == TRUE.

`cv_repeats` The number of repeats during cross-validation. Default: 2; only relevant when cv == TRUE.

`parallel` Indiciate whether the model training will occur parallelized. Default: FALSE. TRUE only possible for non-windows OS.

`n_cores` The number of cores used for parallelized model training via cross-validation. Default: 4. Only relevant on non-windows OS.

`ignore_errors` Indiciate whether errors during model training by caret should be ignored such that another model training try will be initiated until model is trained without raising errors. Default: FALSE.

`known` Indicate whether the true active ligand for a particular dataset is known or not. Default: TRUE. The true ligand will be extracted from the $ligand slot of the setting.

`filter_genes` Indicate whether 50 per cent of the genes that are the least variable in ligand-target scores should be removed in order to reduce the training of the model. Default: FALSE.

Value

A data.frame with for each ligand - data set combination, feature importance scores indicating how important the query ligand is for the prediction of the response in the particular dataset, when prediction is done via a trained regression model with all possible ligands as input. In addition to the importance score(s), the name of the particular setting ($setting), the name of the query ligand($test_ligand), the name of the true active ligand (if known: $ligand).

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation_regression)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances_lm <- dplyr::bind_rows(lapply(settings_ligand_pred, get_multi_ligand_importances_regression,
print(head(ligand_importances_lm))

## End(Not run)
```

get_multi_ligand_rf_importances *Get ligand importances from a multi-ligand trained random forest model.*

Description

get_multi_ligand_rf_importances A random forest is trained to construct one model based on the target gene predictions of all ligands of interest (ligands are considered as features) in order to predict the observed response in a particular dataset. Variable importance scores that indicate for each ligand the importance for response prediction, are extracted. It can be assumed that ligands with higher variable importance scores are more likely to be a true active ligand.

Usage

```
get_multi_ligand_rf_importances(setting,ligand_target_matrix, ligands_position = "cols", ntrees = 100
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) of which the predictve performance need to be assessed; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest. $ligand: NULL or the name of the ligand(s) that are known to be active in the setting of interest.
ligand_target_matrix	A matrix of ligand-target probabilty scores (recommended) or discrete target assignments (not-recommended).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
ntrees	Indicate the number of trees used in the random forest algorithm. The more trees, the longer model training takes, but the more robust the extraced importance scores will be. Default: 1000. Recommended for robustness to have till 10000 trees.
mtry	n**(1/mtry) features of the n features will be sampled at each split during the training of the random forest algorithm. Default: 2 (square root).
continuous	Indicate whether during training of the model, model training and evaluation should be done on class probabilities or discrete class labels. For huge class imbalance, we recommend setting this value to TRUE. Default: TRUE.
known	Indicate whether the true active ligand for a particular dataset is known or not. Default: TRUE. The true ligand will be extracted from the $ligand slot of the setting.
filter_genes	Indicate whether 50 per cent of the genes that are the least variable in ligand-target scores should be removed in order to reduce the training of the model. Default: FALSE.

Value

A data.frame with for each ligand - data set combination, feature importance scores indicating how important the query ligand is for the prediction of the response in the particular dataset, when prediction is done via a trained classification model with all possible ligands as input. In addition to the importance score(s), the name of the particular setting ($setting), the name of the query ligand($test_ligand), the name of the true active ligand (if known: $ligand).

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances_rf <- dplyr::bind_rows(lapply(settings_ligand_pred, get_multi_ligand_rf_importances, ligand_t
print(head(ligand_importances_rf))

## End(Not run)
```

---

get_multi_ligand_rf_importances_regression

Get ligand importances from a multi-ligand trained random forest regression model.

Description

`get_multi_ligand_rf_importances_regression` A random forest is trained to construct one model based on the target gene predictions of all ligands of interest (ligands are considered as features) in order to predict the observed response in a particular dataset (response: e.g. absolute values of log fold change). Variable importance scores that indicate for each ligand the importance for response prediction, are extracted. It can be assumed that ligands with higher variable importance scores are more likely to be a true active ligand.

Usage

```
get_multi_ligand_rf_importances_regression(setting,ligand_target_matrix, ligands_position = "cols", r
```

Arguments

`setting`	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) of which the predictve performance need to be assessed; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest. $ligand: NULL or the name of the ligand(s) that are known to be active in the setting of interest.

ligand_target_matrix
: A matrix of ligand-target probabilty scores (recommended) or discrete target assignments (not-recommended).

ligands_position
: Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

ntrees
: Indicate the number of trees used in the random forest algorithm. The more trees, the longer model training takes, but the more robust the extraced importance scores will be. Default: 1000. Recommended for robustness to have till 10000 trees.

mtry
: n**(1/mtry) features of the n features will be sampled at each split during the training of the random forest algorithm. Default: 2 (square root).

known
: Indicate whether the true active ligand for a particular dataset is known or not. Default: TRUE. The true ligand will be extracted from the $ligand slot of the setting.

filter_genes
: Indicate whether 50 per cent of the genes that are the least variable in ligand-target scores should be removed in order to reduce the training of the model. Default: FALSE.

Value

A data.frame with for each ligand - data set combination, feature importance scores indicating how important the query ligand is for the prediction of the response in the particular dataset, when prediction is done via a trained regression model with all possible ligands as input. In addition to the importance score(s), the name of the particular setting ($setting), the name of the query ligand($test_ligand), the name of the true active ligand (if known: $ligand).

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation_regression)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances_rf <- dplyr::bind_rows(lapply(settings_ligand_pred, get_multi_ligand_rf_importances_regressi
print(head(ligand_importances_rf))

## End(Not run)
```

---

get_ncitations_genes *Get the number of times of gene is mentioned in the pubmed literature*

Description

get_ncitations_genes: Get the number of times of gene is mentioned in the pubmed literature

Usage

```
get_ncitations_genes(file = "ftp://ftp.ncbi.nih.gov/gene/DATA/gene2pubmed.gz")
```

Arguments

file	A file containing a data frame denoting the pubmed ids in which a particular gene entrez of a particulat species is mentioned (variables: taxid, entrez, pubmedid). Default = "ftp://ftp.ncbi.nih.gov/gene/DATA/gene2pubmed.gz"

Value

A data.frame with following variables: entrez, ncitations, symbol,entrez_mouse, symbol_mouse

Examples

```
## Not run:
ncitations <- get_ncitations_genes(file = "ftp://ftp.ncbi.nih.gov/gene/DATA/gene2pubmed.gz")

## End(Not run)
```

---

get_non_spatial_de *Makes a table similar to the output of 'calculate_spatial_DE' and 'process_spatial_de', but now in case you don't have spatial information for the sender and/or receiver celltype. This is needed for comparability reasons.*

Description

get_non_spatial_de Makes a table similar to the output of 'calculate_spatial_DE' and 'process_spatial_de', but now in case you don't have spatial information for the sender and/or receiver celltype. This is needed for comparability reasons.

Usage

```
get_non_spatial_de(niches, spatial_info, type, lr_network)
```

Arguments

niches	a list of lists/niches giving the name, senders and receiver celltypes for each nice. Sender and receiver cell types should be part of Idents(seurat_obj).
spatial_info	Tibble giving information about which celltypes should be compared to each other for defining spatial differential expression. Contains the columns "celltype_region_oi", "celltype_other_region", "niche", "celltype_type".
type	For what type of cellype is the DE analysis: "sender" or "receiver"?
lr_network	Ligand-Receptor Network in tibble format: ligand, receptor as columns

Value

A tibble of mock processed spatial DE information in case you don't have spatial information for the sender and/or receiver celltype.

Examples

```
## Not run:
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
spatial_info <- tibble(
  celltype_region_oi = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
 celltype_other_region = c("LSECs_central", "Hepatocytes_central", "Stellate cells_central")
) %>%
  mutate(niche = "KC_niche", celltype_type = "sender")
get_non_spatial_de(niches, spatial_info, type = "receiver", lr_network)

## End(Not run)
```

---

get_optimized_parameters_nsga2r

Get optimized parameters from the output of `run_nsga2R_cluster`.

Description

`get_optimized_parameters_nsga2` will take as input the output of `run_nsga2R_cluster` and extract the optimal parameter values, either from the best solution at the end of the generations or the best solution across all generations.

Usage

```
get_optimized_parameters_nsga2(result_nsga2r, source_names, search_all_iterations = FALSE, top_n = NUL
```

Arguments

result_nsga2r	The output of run_nsga2R_cluster.
source_names	Character vector containing the names of the data sources.
search_all_iterations	Logical indicating whether the best solution across all generations should be considered (TRUE) or only the best solution at the end of the generations (FALSE).
top_n	If search_all_iterations=TRUE, this indicates how many of the best solutions should be considered.
summarise_weights	If search_all_iterations=TRUE, a logical indicating whether the weights should be summarised by taking the mean and median.

Value

A list containing two dataframes, the optimal data source weights and the optimal hyperparameters.

Examples

```
## Not run:
results <- run_nsga2R_cluster(model_evaluation_optimization_nsga2r,
  varNo = n_param, objDim = n_obj,
 lowerBounds = lower_bounds, upperBounds = upper_bounds, popSize = 360, tourSize = 2, generations = 15, ncores = 8
)

# Get the best solution at the end of the generations
optimized_parameters <- get_optimized_parameters_nsga2(results, source_names, search_all_iterations = FALSE, top_

# Get the best solution across all generations, consider top 25 solutions and summarise weights
optimized_parameters <- get_optimized_parameters_nsga2(results, source_names, search_all_iterations = TRUE, top_n

## End(Not run)
```

---

get_prioritization_tables

Use the information from the niche- and spatial differential expression analysis of ligand-senders and receptor-receivers pairs, in addition to the ligand activity prediction and ligand-target inferernce, in order to make a final ligand-receptor and ligand-target prioritization table.

Description

get_prioritization_tables Use the information from the niche- and spatial differential expression analysis of ligand-senders and receptor-receivers pairs, in addition to the ligand activity prediction and ligand-target inferernce, in order to make a final ligand-receptor and ligand-target prioritization table.

Usage

```
get_prioritization_tables(output_nichenet_analysis, prioritizing_weights)
```

Arguments

output_nichenet_analysis	List containing following data frames: DE_sender_receiver, ligand_scaled_receptor_expression_fraction_ sender_spatial_DE_processed, receiver_spatial_DE_processed, ligand_activities_targets, DE_receiver_processed_targets, exprs_tbl_ligand, exprs_tbl_receptor, exprs_tbl_target
prioritizing_weights	Named numeric vector in the form of: #' prioritizing_weights = c("scaled_ligand_score" = 5,

Value

A list containing a prioritization table for ligand-receptor interactions, and one for ligand-target interactions

Examples

```
## Not run:
prioritizing_weights <- c(
  "scaled_ligand_score" = 5,
  "scaled_ligand_expression_scaled" = 1,
  "ligand_fraction" = 1,
  "scaled_ligand_score_spatial" = 2,
  "scaled_receptor_score" = 0.5,
  "scaled_receptor_expression_scaled" = 0.5,
  "receptor_fraction" = 1,
  "ligand_scaled_receptor_expression_fraction" = 1,
  "scaled_receptor_score_spatial" = 0,
  "scaled_activity" = 0,
  "scaled_activity_normalized" = 1
)
output_nichenet_analysis <- list(
 DE_sender_receiver = DE_sender_receiver, ligand_scaled_receptor_expression_fraction_df = ligand_scaled_receptor
 ligand_activities_targets = ligand_activities_targets, DE_receiver_processed_targets = DE_receiver_processed_ta
)
prioritization_tables <- get_prioritization_tables(output_nichenet_analysis, prioritizing_weights)

## End(Not run)
```

---

get_single_ligand_importances

Get ligand importances based on target gene prediction performance of single ligands.

Description

get_single_ligand_importances Get ligand importance measures for ligands based on how well a single, individual, ligand can predict an observed response. Assess how well every ligand of interest is able to predict the observed transcriptional response in a particular dataset, according to the ligand-target model. It can be assumed that the ligand that best predicts the observed response, is more likely to be the true ligand.

Usage

```
get_single_ligand_importances(setting,ligand_target_matrix, ligands_position = "cols", known = TRUE)
```

Arguments

setting	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) of which the predictve performance need to be assessed; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest. $ligand: NULL or the name of the ligand(s) that are known to be active in the setting of interest.
ligand_target_matrix	A matrix of ligand-target probabilty scores (or discrete target assignments).
ligands_position	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
known	Indicate whether the true active ligand for a particular dataset is known or not. Default: TRUE. The true ligand will be extracted from the $ligand slot of the setting.

Value

A data.frame with for each ligand - data set combination, classification evaluation metrics indicating how well the query ligand predicts the response in the particular dataset. Evaluation metrics are the same as in evaluate_target_prediction. In addition to the metrics, the name of the particular setting ($setting), the name of the query ligand($test_ligand), the name of the true active ligand (if known: $ligand).

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
print(head(ligand_importances))

## End(Not run)
```

`get_single_ligand_importances_regression` *Get ligand importances based on target gene value prediction performance of single ligands (regression).*

Description

`get_single_ligand_importances_regression` Get ligand importance measures for ligands based on how well a single, individual, ligand can predict an observed response. Assess how well every ligand of interest is able to predict the observed transcriptional response in a particular dataset, according to the ligand-target model. It can be assumed that the ligand that best predicts the observed response, is more likely to be the true ligand. Response: continuous values associated to a gene, e.g. a log fold change value.

Usage

```
get_single_ligand_importances_regression(setting,ligand_target_matrix, ligands_position = "cols", kno
```

Arguments

`setting`	A list containing the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) of which the predictve performance need to be assessed; .$response: the observed target response: indicate for a gene whether it was a target or not in the setting of interest. $ligand: NULL or the name of the ligand(s) that are known to be active in the setting of interest.
`ligand_target_matrix`	A matrix of ligand-target probabilty scores (or discrete target assignments).
`ligands_position`	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"
`known`	Indicate whether the true active ligand for a particular dataset is known or not. Default: TRUE. The true ligand will be extracted from the $ligand slot of the setting.

Value

A data.frame with for each ligand - data set combination, regression model fit metrics indicating how well the query ligand predicts the response in the particular dataset. Evaluation metrics are the same as in `evaluate_target_prediction_regression`. In addition to the metrics, the name of the particular setting ($setting), the name of the query ligand($test_ligand), the name of the true active ligand (if known: $ligand).

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation_regression)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s
```

```
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances_regression, li
print(head(ligand_importances))

## End(Not run)
```

`get_slope_ligand_popularity` *Regression analysis between ligand popularity and target gene predictive performance*

Description

`get_slope_ligand_popularity`: Performs regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of the ligand.

Usage

```
get_slope_ligand_popularity(metric, performances)
```

Arguments

`metric`	The name of the performance metric of which the trend with the popularity of the ligand should be calculated.
`performances`	A data.frame in which the performance measures for target gene predictions of ligands are denoted together with the popularity of the ligand. (should contain at least following variables: ligand, ncitations and the metric of interest)

Value

A data.frame in which the regression coefficient estimate, p-value and corresponding R-squared value are shown for the regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of the ligand.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:10], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
performances <- bind_rows(lapply(settings, evaluate_target_prediction, ligand_target_matrix))
# ncitations = get_ncitations_genes()
performances_ligand_popularity <- add_ligand_popularity_measures_to_perfs(performances, ncitations)
slopes_auroc <- get_slope_ligand_popularity("auroc", performances_ligand_popularity)
```

```
slopes_df <- performances_ligand_popularity %>%
  select(-setting, -ligand, -ncitations) %>%
  colnames() %>%
  lapply(., get_slope_ligand_popularity, performances_ligand_popularity) %>%
  bind_rows()

## End(Not run)
```

`get_slope_target_gene_popularity` *Regression analysis between target gene popularity and target gene predictive performance*

Description

`get_slope_target_gene_popularity`: Performs regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of target genes.

Usage

```
get_slope_target_gene_popularity(metric,performances,method = "individual")
```

Arguments

`metric`	The name of the performance metric of which the trend with the popularity of the target genes should be calculated.
`performances`	A data.frame in which the performance measures for target gene predictions of ligands are denoted together with the popularity bin of the target genes for which predictions were evaluated (should contain at least following variables: target_bin_id and the metric of interest)
`method`	'All': calculate slope by considering all datasets in settings. 'Individual': calculate slope for every dataset in settings separately to investigate dataset-specific popularity bias. Default: 'individual'.

Value

A data.frame in which the regression coefficient estimate, p-value and corresponding R-squared value are shown for the regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of the target genes.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:10], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(settings)
```

```
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
# ncitations = get_ncitations_genes()
performances_target_bins_popularity <- evaluate_target_prediction_per_bin(5, settings, ligand_target_matrix, nci
slopes_auroc <- get_slope_target_gene_popularity("auroc", performances_target_bins_popularity)
slopes_df <- performances_target_bins_popularity %>%
  select(-setting, -ligand, -target_bin_id) %>%
  colnames() %>%
 lapply(., get_slope_target_gene_popularity, performances_target_bins_popularity, method = "individual") %>%
  bind_rows()
slopes_df2 <- performances_target_bins_popularity %>%
  select(-setting, -ligand, -target_bin_id) %>%
  colnames() %>%
 lapply(., get_slope_target_gene_popularity, performances_target_bins_popularity, method = "all") %>%
  bind_rows()

## End(Not run)
```

`get_slope_target_gene_popularity_ligand_prediction`
Regression analysis between target gene popularity and ligand activity predictive performance

---

Description

`get_slope_target_gene_popularity_ligand_prediction`: Performs regression analysis to investigate the trend between a particular classficiation evaluation metric (ligand activity prediction) and the popularity of target genes.

Usage

```
get_slope_target_gene_popularity_ligand_prediction(metric,performances)
```

Arguments

`metric`	The name of the performance metric of which the trend with the popularity of the target genes should be calculated.
`performances`	A data.frame in which the performance measures for ligand activity predictions of ligands are denoted together with the popularity bin of the target genes for which predictions were evaluated (should contain at least following variables: target_bin_id and the metric of interest)

Value

A data.frame in which the regression coefficient estimate, p-value and corresponding R-squared value are shown for the regression analysis to investigate the trend between a particular classficiation evaluation metric and the popularity of the target genes.

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:10], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
# ncitations = get_ncitations_genes()
performances_target_bins_popularity <- evaluate_ligand_prediction_per_bin(5, settings, ligand_target_matrix, nci
slopes_auroc <- get_slope_target_gene_popularity_ligand_prediction("auroc", performances_target_bins_popularity
slopes_df <- performances_target_bins_popularity %>%
  select(-importance_measure, -target_bin_id) %>%
  colnames() %>%
 lapply(., get_slope_target_gene_popularity_ligand_prediction, performances_target_bins_popularity) %>%
  bind_rows()

## End(Not run)
```

---

get_target_genes_ligand_oi

Get a set of predicted target genes of a ligand of interest

Description

get_target_genes_ligand_oi Get a set of predicted target genes of a ligand of interest.

Usage

```
get_target_genes_ligand_oi(ligand_oi, ligand_target_matrix, error_rate = 0.1, cutoff_method = "distrib
```

Arguments

ligand_oi	The ligand of interest of which top target genes should be returned
ligand_target_matrix	A matrix of ligand-target probabilty scores.
error_rate	FDR for cutoff_method "fdrtool" and "distribution"; number between 0 and 1 indicating which top fraction of target genes should be returned for cutoff_method "quantile". Default: 0.1
cutoff_method	Method to determine which genes can be considered as a target of a ligand and which genes not, based on the ligand-target probability scores. Possible options: "distribution", "fdrtool" and "quantile". Default: "distribution".
fdr_method	Only relevant when cutoff_method is "fdrtool". Possible options: "global" and "local". Default: "global".
output	Determines whether a vector with target gene names should be returned ("gene_symbols") or a logical vector indicating for every target gene whether or not it is a target ("logical").

ligands_position
: Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A vector with target gene names should be returned ("gene_symbols") or a logical vector indicating for every target gene whether or not it is a target ("logical").

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
targets <- get_target_genes_ligand_oi("BMP2", ligand_target_matrix, error_rate = 0.1, cutoff_method = "distributi

## End(Not run)
```

---

get_top_predicted_genes

Find which genes were among the top-predicted targets genes in a specific cross-validation round and see whether these genes belong to the gene set of interest as well.

Description

get_top_predicted_genes Find which genes were among the top-predicted targets genes in a specific cross-validation round and see whether these genes belong to the gene set of interest as well.

Usage

```
get_top_predicted_genes(round,gene_prediction_list, quantile_cutoff = 0.95)
```

Arguments

round
: Integer describing which fold of the cross-validation scheme it is.

gene_prediction_list
: List with per round of cross-validation: a tibble with columns "gene", "prediction" and "response" (e.g. output of function ‘assess_rf_class_probabilities‘)

quantile_cutoff
: Quantile of which genes should be considered as top-predicted targets. Default: 0.95, thus considering the top 5 percent predicted genes as predicted targets.

Value

A tibble indicating for every gene whether it belongs to the geneset and whether it belongs to the top-predicted genes in a specific cross-validation round.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
geneset <- c("SOCS2", "SOCS3", "IRF1")
background_expressed_genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
gene_predictions_list <- seq(2) %>% lapply(assess_rf_class_probabilities, 2, geneset = geneset, background_expres
seq(length(gene_predictions_list)) %>% lapply(get_top_predicted_genes, gene_predictions_list)

## End(Not run)
```

---

get_weighted_ligand_receptor_links

Get the weighted ligand-receptor links between a possible ligand and its receptors

---

Description

`get_weighted_ligand_receptor_links` Get the weighted ligand-receptor links between a possible ligand and its receptors

Usage

```
get_weighted_ligand_receptor_links(
  best_upstream_ligands,
  expressed_receptors,
  lr_network,
  weighted_networks_lr_sig
)
```

Arguments

`best_upstream_ligands`	Character vector containing ligands of interest.
`expressed_receptors`	Character vector of receptors expressed in the cell type of interest.
`lr_network`	A data frame with two columns, `from` and `to`, containing the ligand-receptor interactions.
`weighted_networks_lr_sig`	A data frame with three columns, `from`, `to` and `weight`, containing the ligand-receptor interactions and their weights.

Examples

```
## Not run:
ligand_receptor_links_df <- get_weighted_ligand_receptor_links(best_upstream_ligands, expressed_receptors, lr_ne

## End(Not run)
```

`get_weighted_ligand_target_links`

Infer weighted active ligand-target links between a possible ligand and target genes of interest

---

Description

`get_weighted_ligand_target_links` Infer active ligand target links between possible ligands and genes belonging to a gene set of interest: consider the intersect between the top n targets of a ligand and the gene set.

Usage

```
get_weighted_ligand_target_links(ligand, geneset,ligand_target_matrix,n = 250)
```

Arguments

`ligand`	Character vector giving the gene symbols of the potentially active ligand for which you want to find target genes.
`geneset`	Character vector of the gene symbols of genes of which the expression is potentially affected by ligands from the interacting cell.
`ligand_target_matrix`	The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).
`n`	The top n of targets per ligand that will be considered. Default: 250.

Value

A tibble with columns ligand, target and weight (i.e. regulatory potential score).

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligand <- "TNF"
geneset <- c("SOCS2", "SOCS3", "IRF1")
active_ligand_target_links_df <- get_weighted_ligand_target_links(ligand = potential_ligand, geneset = geneset, l

## End(Not run)
```

hyperparameter_list *Optimized hyperparameter values*

Description

A list

Usage

```
hyperparameter_list
```

Format

A list

lr_sig_hub ligand-signaling network hub correction factor

gr_hub gene regulatory network hub correction factor

ltf_cutoff cutoff on PPR scores

damping_factor damping factor used in the PPR algorithm

infer_supporting_datasources *Get the data sources that support the specific interactions in the extracted ligand-target signaling subnetwork*

Description

`infer_supporting_datasources` Get the data sources that support the specific interactions in the extracted ligand-target signaling subnetwork

Usage

```
infer_supporting_datasources(signaling_graph_list,lr_network, sig_network, gr_network)
```

Arguments

`signaling_graph_list`	A list of two elements: sig: a data frame/ tibble containg weighted ligand-receptor and signaling interactions (from, to, weight); and gr: a data frame/tibble containng weighted gene regulatory interactions (from, to, weight)
`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)
`gr_network`	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)

Value

A tibble with columns from, to, source and layer

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_tf_matrix <- construct_ligand_tf_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99, algorit
all_ligands <- c("BMP2")
all_targets <- c("HEY1")
top_n_regulators <- 2
ligand_target_signaling_list <- get_ligand_signaling_path(ligand_tf_matrix, all_ligands, all_targets, top_n_regu
data_source_info_network <- infer_supporting_datasources(ligand_target_signaling_list, lr_network, sig_network,

## End(Not run)
```

---

ligand_activity_performance_top_i_removed

Calculate ligand activity performance without considering evaluation datasets belonging to the top i most frequently cited ligands

---

Description

ligand_activity_performance_top_i_removed: calculates ligand activity performance (given ligand importances measures) without considering evaluation datasets belonging to the top i most frequently cited ligands.

Usage

```
ligand_activity_performance_top_i_removed(i, importances, ncitations)
```

Arguments

i	Indicate the number of most popular ligands for which the corresponding datasets will be removed for performance calculation.
importances	Data frame containing the ligand importance measures for every ligand-dataset combination. See get_single_ligand_importances and get_multi_ligand_importances.
ncitations	A data frame denoting the number of times a gene is mentioned in the Pubmed literature. Should at least contain following variables: 'symbol' and 'ncitations'. Default: ncitations (variable contained in this package). See function get_ncitations_genes for a function that makes this data frame from current Pubmed information.

Value

A data.frame in which you can find several evaluation metrics of the ligand activity prediction performance and an indication of what percentage of most popular ligands has not been considered ($popularity_index).

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target

ligand_activity_popularity_bias <- lapply(0:3, ligand_activity_performance_top_i_removed, ligand_importances, nc
# example plot
# ligand_activity_popularity_bias %>% ggplot(aes(popularity_index,aupr)) + geom_smooth(method = "lm") + geom_poin

## End(Not run)
```

---

make_circos_lr *make_circos_lr*

Description

make_circos_lr Plot the prioritized ligand-receptor pairs in a circos plot (via the circlize package)

Usage

```
make_circos_lr(
  prioritized_tbl_oi,
  colors_sender,
  colors_receiver,
  cutoff = 0,
  scale = FALSE,
  transparency = NULL,
  circos_type = "normal",
  border = TRUE,
  separate_legend = FALSE
)
```

Arguments

prioritized_tbl_oi	Dataframe with the ligand-receptor interactions that should be visualized
colors_sender	Named character vector giving the colors of each sender cell type
colors_receiver	Named character vector giving the colors of each receiver cell type
cutoff	Threshold On the prioritization score - if lower than this value, the link will be removed – default = 0.

scale	scale value in ‘chordDiagram‘. Default: FALSE
transparency	Vector of transparency values of the links or NULL, in that case this will be calculated automatically. Default: NULL.
circos_type	"normal" or "arrow". Default: "normal".
border	Border to arrows or not in ‘chordDiagram‘? (Default: TRUE)
separate_legend	return plot and legend as separate objects? (Default: FALSE)

Value

List containing the circos plot and the legend

Examples

```
## Not run:
make_circos_lr(prioritized_tbl_oi, colors_sender, colors_receiver, cutoff = 0, scale = FALSE, transparency = NULL,

## End(Not run)
```

---

make_circos_plot *Draw a circos plot*

Description

Draw a circos plot

Usage

```
make_circos_plot(vis_circos_obj, transparency = FALSE, args.circos.text = list(), ...)
```

Arguments

vis_circos_obj	Object returned by prepare_circos_visualization
transparency	Logical indicating whether the transparency of the links will correspond to the ligand-target potential score (default: FALSE)
args.circos.text	List of arguments to pass to circos.text (by default, the text size is set to 1)
...	Additional arguments to pass to chordDiagram

Value

A circos plot

Examples

```
## Not run:
# Default
make_circos_plot(vis_circos_obj, transparency = FALSE)

# Transparency
make_circos_plot(vis_circos_obj, transparency = TRUE)

# Make text smaller
make_circos_plot(vis_circos_obj, transparency = TRUE, args.circos.text = list(cex = 0.5))

# Don't sort links of each ligand based on widths (not recommended)
make_circos_plot(vis_circos_obj, transparency = TRUE, args.circos.text = list(cex = 0.5), link.sort = FALSE)

## End(Not run)
```

`make_discrete_ligand_target_matrix` *Convert probabilistic ligand-target matrix to a discrete one.*

Description

`make_discrete_ligand_target_matrix` Convert probabilistic ligand-target matrix to a discrete one. This means that for every ligand, genes will be labeled as target (TRUE) or non-target (FALSE).

Usage

```
make_discrete_ligand_target_matrix(ligand_target_matrix, error_rate = 0.1, cutoff_method = "distribut
```

Arguments

`ligand_target_matrix`	A matrix of ligand-target probabilty scores.
`error_rate`	FDR for cutoff_method "fdrtool" and "distribution"; number between 0 and 1 indicating which top fraction of target genes should be returned for cutoff_method "quantile". Default: 0.1
`cutoff_method`	Method to determine which genes can be considered as a target of a ligand and which genes not, based on the ligand-target probability scores. Possible options: "distribution", "fdrtool" and "quantile". Default: "distribution".
`fdr_method`	Only relevant when cutoff_method is "fdrtool". Possible options: "global" and "local". Default: "global".
`ligands_position`	Indicate whether the ligands in the ligand-target matrix are in the rows ("rows") or columns ("cols"). Default: "cols"

Value

A matrix of ligand-target assignments. TRUE: gene is a target of the ligand of interest; FALSE: gene is not a target of the ligand of interest.

Examples

```
## Not run:
## Generate the ligand-target matrix from loaded weighted_networks
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_target_matrix <- make_discrete_ligand_target_matrix(ligand_target_matrix, error_rate = 0.1, cutoff_method

## End(Not run)
```

---

make_heatmap_bidir_lt_ggplot

Make a ggplot heatmap object from an input ligand-target matrix.

Description

make_heatmap_bidir_lt_ggplot Make a ggplot heatmap object from an input ligand-target matrix in which it is indicated whether a gene is a top target of a ligand ("top-target"), the ligand is a top ligand of the gene ("top-ligand") or both ("top") or none ("none").

Usage

```
make_heatmap_bidir_lt_ggplot(matrix, y_name, x_name, y_axis = TRUE, x_axis = TRUE, x_axis_position = "t
#'
```

Arguments

matrix	Matrix with continuous values to plot in heatmap
y_name	Title of the y-axis
x_name	Title of the x-axis
y_axis	Should y-axis label names and titles be displayed? TRUE or FALSE. Default: TRUE.
x_axis	Should x-axis label names and titles be displayed? TRUE or FALSE. Default: TRUE.
x_axis_position	X-axis position: "top" or "bottomm"; only relevant if x_axis == TRUE. Default:"top".
legend_position	Legend position: "top", "bottom", "left", "right" or "none". Default: "top"
...	Optional arguments passed to element_text(); used to set font type and size of axis labels and titles.

Value

A ggplot object displaying a heatmap

Examples

```
## Not run:
p <- make_heatmap_bidir_lt_ggplot(bidirectional_ligand_target_matrix_vis, y_name = "ligand", x_name = "target")

## End(Not run)
```

`make_heatmap_ggplot` *Make a ggplot heatmap object from an input matrix (2-color).*

Description

`make_heatmap_ggplot` Make a ggplot heatmap object from an input matrix containing continuous values. Two-color scale from white to color of choice.

Usage

```
make_heatmap_ggplot(matrix, y_name, x_name, y_axis = TRUE,x_axis = TRUE, x_axis_position = "top", legen
```

Arguments

`matrix`	Matrix with continuous values to plot in heatmap
`y_name`	Title of the y-axis
`x_name`	Title of the x-axis
`y_axis`	Should y-axis label names and titles be displayed? TRUE or FALSE. Default: TRUE.
`x_axis`	Should x-axis label names and titles be displayed? TRUE or FALSE. Default: TRUE.
`x_axis_position`	X-axis position: "top" or "bottomm"; only relevant if x_axis == TRUE. Default:"top".
`legend_position`	Legend position: "top", "bottom", "left", "right" or "none". Default: "top".
`color`	Color for highest continuous value in heatmap. Color gradient will go from "whitesmoke" to this color. Default: "blue".
`legend_title`	Title of the legend.
`...`	Optional arguments passed to element_text(); used to set font type and size of axis labels and titles.

Value

A ggplot object displaying a heatmap

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99,
p <- make_heatmap_ggplot(ligand_target_matrix[1:50, ] %>% t(), y_name = "ligand", x_name = "target")

## End(Not run)
```

---

make_ligand_activity_target_exprs_plot

make_ligand_activity_target_exprs_plot

Description

make_ligand_activity_target_exprs_plot Plot the ligand expression in senders plus their activity and target genes in receivers

Usage

```
make_ligand_activity_target_exprs_plot(receiver_oi, prioritized_tbl_oi, prioritization_tbl_ligand_re
exprs_tbl_ligand, exprs_tbl_target, lfc_cutoff, ligand_target_matrix, scaled_ligand_activity_limits =
```

Arguments

receiver_oi	Name of the receiver cell type of interest
prioritized_tbl_oi	Dataframe with the ligand-receptor interactions that should be visualized
prioritization_tbl_ligand_receptor	$prioritization_tbl_ligand_receptor slot of 'get_prioritization_tables'
prioritization_tbl_ligand_target	$prioritization_tbl_ligand_target slot of 'get_prioritization_tables'
exprs_tbl_ligand	Dataframe with the expression values for the ligands in the sender cell types
exprs_tbl_target	Dataframe with the expression values for the targets in the receiver cell types
lfc_cutoff	Cutoff used on the logFC value
ligand_target_matrix	ligand-target matrix
scaled_ligand_activity_limits	limits used in the heatmap for the scaled ligand activity, one of "abs_max" (-+ absolute maximum), "min_max" (minimum and maximum), or "IQR" (cf. boxplot, outliers are squished to the limits)
plot_legend	TRUE (default): add legend to the plot. FALSE: do not add legend.

heights — automatic determination if default NULL. If not NULL: number given by the user to indicate the requested heights, which are the height proportions of the different row panels in the plot.

widths — automatic determination if default NULL. If not NULL: number given by the user to indicate the requested widths, which are the width proportions of the different columns (side-by-side heatmaps) in the plot.

Value

List containing the combined heatmap and the legend

Examples

```
## Not run:
make_ligand_activity_target_exprs_plot(receiver_oi, prioritized_tbl_oi, prioritization_tbl_ligand_receptor, pri

## End(Not run)
```

`make_ligand_receptor_lfc_plot`
make_ligand_receptor_lfc_plot

Description

`make_ligand_receptor_lfc_plot` Plot the ligand logFC in senders plus the logFC of their receptors in the receivers.

Usage

```
make_ligand_receptor_lfc_plot(receiver_oi, prioritized_tbl_oi, prioritization_tbl_ligand_receptor, p
```

Arguments

receiver_oi — Name of the receiver cell type of interest

prioritized_tbl_oi — Dataframe with the ligand-receptor interactions that should be visualized

prioritization_tbl_ligand_receptor — $prioritization_tbl_ligand_receptor slot of 'get_prioritization_tables'

plot_legend — TRUE (default): add legend to the plot. FALSE: do not add legend.

heights — automatic determination if default NULL. If not NULL: number given by the user to indicate the requested heights, which are the height proportions of the different row panels in the plot.

widths — automatic determination if default NULL. If not NULL: number given by the user to indicate the requested widths, which are the width proportions of the different columns (side-by-side heatmaps) in the plot.

Value

List containing the combined heatmap and the legend

Examples

```
## Not run:
make_ligand_receptor_lfc_plot(receiver_oi, prioritized_tbl_oi, prioritization_tbl_ligand_receptor, plot_legend =

## End(Not run)
```

`make_ligand_receptor_lfc_spatial_plot`
make_ligand_receptor_lfc_spatial_plot

Description

`make_ligand_receptor_lfc_spatial_plot` Plot the ligand logFC in senders plus the logFC of their receptors in the receivers. In addition, add the spatialDE logFC values of the ligands!

Usage

```
make_ligand_receptor_lfc_spatial_plot(receiver_oi, prioritized_tbl_oi, prioritization_tbl_ligand_rec
```

Arguments

`receiver_oi`	Name of the receiver cell type of interest
`prioritized_tbl_oi`	Dataframe with the ligand-receptor interactions that should be visualized
`prioritization_tbl_ligand_receptor`	$prioritization_tbl_ligand_receptor slot of 'get_prioritization_tables'
`ligand_spatial`	TRUE if need to plot the ligand spatial DE info, FALSE if not.
`receptor_spatial`	TRUE if need to plot the receptor spatial DE info, FALSE if not.
`plot_legend`	TRUE (default): add legend to the plot. FALSE: do not add legend.
`heights`	automatic determination if default NULL. If not NULL: number given by the user to indicate the requested heights, which are the height proportions of the different row panels in the plot.
`widths`	automatic determination if default NULL. If not NULL: number given by the user to indicate the requested widths, which are the width proportions of the different columns (side-by-side heatmaps) in the plot.

Value

List containing the combined heatmap and the legend

Examples

```
## Not run:
make_ligand_receptor_lfc_spatial_plot(receiver_oi, prioritized_tbl_oi, prioritization_tbl_ligand_receptor, liga

## End(Not run)
```

`make_line_plot` *Make a line plot*

Description

Make a line plot comparing the ranking of ligands based on their activities in the sender-agnostic and sender-focused approaches

Usage

```
make_line_plot(ligand_activities, potential_ligands, ranking_range = c(1, 20), agnostic_color = "tomat
```

Arguments

`ligand_activities`	Dataframe containing the ligand activities from the sender-agnostic approach
`potential_ligands`	Character vector containing the ligands that are expressed in the sender cell type (i.e. the ligands that are used in the sender-focused approach)
`ranking_range`	Numeric vector of length 2 indicating the range of the rankings to be displayed (default: c(1, 20))
`agnostic_color`	Color representing ligands only inthe sender-agnostic approach (default: "tomato")
`focused_color`	Color representing expressed ligands from the sender-focused approach (default: "black")
`tied_color`	Color to shade ligands that are tied in the same rank (default: "gray75")
`inset_scale`	Numeric value indicating the size of the points and text in the inset (default: 1)
`label_size`	Numeric value indicating size of the data labels, before scaling (default: 3.88, same as older versions, but now explicitly settable)
`point_size`	Numeric value indicating size of the data points, before scaling (default: 1.5, same as older versions, but now explicitly settable)
`line_width`	Numeric value indicating size of the line width, before scaling (default: 0.5, same as older versions, but now explicitly settable)

Value

A ggplot object showing the distribution of sender-focused ligands, as well as a line plot inset comparing the rankings between the two approaches

Examples

```
## Not run:
# Default
make_line_plot(ligand_activities, potential_ligands)

## End(Not run)
```

`make_mushroom_plot` *Make a "mushroom plot" of ligand-receptor interactions*

Description

`make_mushroom_plot` Make a plot in which each glyph consists of two semicircles corresponding to ligand- and receptor- information. The size of the semicircle is the percentage of cells that express the protein, while the saturation corresponds to the scaled average expression value.

Usage

```
make_mushroom_plot(
  prioritization_table,
  top_n = 30,
  show_rankings = FALSE,
  show_all_datapoints = FALSE,
  true_color_range = TRUE,
  use_absolute_rank = FALSE,
  size = "scaled_avg_exprs",
  color = "scaled_p_val_adapted",
  ligand_fill_colors = c("#DEEBF7", "#08306B"),
  receptor_fill_colors = c("#FEE0D2", "#A50F15"),
 unranked_ligand_fill_colors = c(alpha("#FFFFFF", alpha = 0.2), alpha("#252525", alpha =
    0.2)),
 unranked_receptor_fill_colors = c(alpha("#FFFFFF", alpha = 0.2), alpha("#252525", alpha
    = 0.2)),
  label_size = 3.88,
  ...
)
```

Arguments

`prioritization_table`	A prioritization table as generated by `generate_prioritization_tables`.
`top_n`	An integer indicating how many ligand-receptor pairs to show
`show_rankings`	A logical indicating whether to show the ranking of the ligand-receptor pairs (default: FALSE)
`show_all_datapoints`	A logical indicating whether to show all ligand-receptor pairs (default: FALSE, if true they will be grayed out)

`true_color_range`
: A logical indicating whether to use the default color range as determined by ggplot (TRUE, default) or set the limits to a range of 0-1 (FALSE)

`use_absolute_rank`
: A logical indicating to whether use the absolute prioritization rank to filter the top_n ligand-receptor pairs (default: FALSE)

`size`
: A string indicating which column to use for the size of the semicircles (default: "scaled_avg_exprs"; use column name without "_ligand" or "_receptor" suffix)

`color`
: A string indicating which column to use for the color of the semicircles (default: "scaled_p_val_adapted"; use column name without "_ligand" or "_receptor" suffix)

`ligand_fill_colors`
: A vector of the low and high colors to use for the ligand semicircle fill gradient (default: c("#DEEBF7", "#08306B"))

`receptor_fill_colors`
: A vector of the low and high colors to use for the receptor semicircle fill gradient (default: c("#FEE0D2", "#A50F15"))

`unranked_ligand_fill_colors`
: A vector of the low and high colors to use for the unranked ligands when show_all_datapoints is TRUE (default: c(alpha("#FFFFFF", alpha=0.2), alpha("#252525", alpha=0.2)))

`unranked_receptor_fill_colors`
: A vector of the low and high colors to use for the unranked receptors when show_all_datapoints is TRUE (default: c(alpha("#FFFFFF", alpha=0.2), alpha("#252525", alpha=0.2)))

`...`
: Additional arguments passed to `ggplot2::theme`. As there are often issues with the scales legend, it is recommended to change legend sizes and positions using this argument, i.e., `legend.key.height`, `legend.key.width`, `legend.title`, and `legend.text`.

Details

If the values range of the column used as the "size" parameter is not between 0 and 1.001, an error will be thrown.

The sender cell types can be ordered by encoding the "sender" column as a factor. If the "sender" column is not a factor, the sender cell types will be ordered alphabetically.

By default, the top_n ligand-receptor pairs are shown despite their absolute ranking. So, if a receiver cell type has LR pairs that are only ranked from 31-40 and the top_n is set to 20, the LR pairs will be shown. If use_absolute_rank is set to TRUE, only LR pairs with absolute ranking from 1-20 will be shown.

Value

A ggplot object

Examples

```
## Not run:
# Create a prioritization table
prior_table <- generate_prioritization_tables(processed_expr_table, processed_DE_table, ligand_activities, proce
make_mushroom_plot(prior_table)


# Show only top 20, and write rankings on the plot
make_mushroom_plot(prior_table, top_n = 20, show_rankings = TRUE)

# Show all datapoints, and use true color range
make_mushroom_plot(prior_table, show_all_datapoints = TRUE, true_color_range = TRUE)


# Change the size and color columns
make_mushroom_plot(prior_table, size = "pct_expressed", color = "scaled_avg_exprs")


# For a prioritization table with multiple receiver cell types
make_mushroom_plot(prior_table_combined %>% filter(receiver == celltype_oi))

## End(Not run)
```

---

make_threecolor_heatmap_ggplot

Make a ggplot heatmap object from an input matrix (3-color).

Description

`make_threecolor_heatmap_ggplot` Make a ggplot heatmap object from an input matrix containing continuous values. Three-color scale with colors of choice. Ideal for plotting log fold change expression.

Usage

```
make_threecolor_heatmap_ggplot(matrix, y_name, x_name, y_axis = TRUE,x_axis = TRUE, x_axis_position = '
```

Arguments

`matrix`	Matrix with continuous values to plot in heatmap
`y_name`	Title of the y-axis
`x_name`	Title of the x-axis
`y_axis`	Should y-axis label names and titles be displayed? TRUE or FALSE. Default: TRUE.
`x_axis`	Should x-axis label names and titles be displayed? TRUE or FALSE. Default: TRUE.

x_axis_position
: X-axis position: "top" or "bottomm"; only relevant if x_axis == TRUE. Default:"top".

legend_position
: Legend position: "top", "bottom", "left", "right" or "none". Default: "top".

low_color
: Color for lowest continuous value in heatmap. Color gradient will go from "whitesmoke" to this color. Default: "blue".

mid_color
: Color for the "mid" value as defined by that parameter. Default: "whitesmoke".

high_color
: Color for highest continuous value in heatmap. Color gradient will go from "whitesmoke" to this color. Default: "red".

mid
: Continuous value that will receive the "mid_color" color. Default: 0

legend_title
: Title of the legend.

...
: Optional arguments passed to element_text(); used to set font type and size of axis labels and titles.

Value

A ggplot object displaying a heatmap

Examples

```
## Not run:
library(dplyr)
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", c("IL4", "IL13"))
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0.99,
p <- make_threecolor_heatmap_ggplot(ligand_target_matrix[1:50, ] %>% t(), y_name = "ligand", x_name = "target")

## End(Not run)
```

---

mlrmbo_optimization *Optimization of objective functions via model-based optimization (mlrMBO).*

Description

`mlrmbo_optimization` will execute multi-objective model-based optimization of an objective function. The defined surrogate learner here is "kriging".

Usage

```
mlrmbo_optimization(run_id,obj_fun,niter,ncores,nstart,additional_arguments)
```

Arguments

run_id	Indicate the id of the optimization run.
obj_fun	An objective function as created by the function mlrMBO::makeMultiObjectiveFunction.
niter	The number of iterations during the optimization process.
ncores	The number of cores on which several parameter settings will be evaluated in parallel.
nstart	The number of different parameter settings used in the begin design.
additional_arguments	A list of named additional arguments that will be passed on the objective function.

Value

A result object from the function mlrMBO::mbo. Among other things, this contains the optimal parameter settings, the output corresponding to every input etc.

Examples

```
## Not run:
library(dplyr)
library(mlrMBO)
library(parallelMap)
additional_arguments_topology_correction <- list(source_names = source_weights_df$source %>% unique(), algorithm
nr_datasources <- additional_arguments_topology_correction$source_names %>% length()

obj_fun_multi_topology_correction <- makeMultiObjectiveFunction(name = "nichenet_optimization", description = "da

mlrmbo_optimization <- lapply(1, mlrmbo_optimization, obj_fun = obj_fun_multi_topology_correction, niter = 3, ncor

## End(Not run)
```

---

model_based_ligand_activity_prediction

Prediction of ligand activity prediction by a model trained on ligand importance scores.

Description

model_based_ligand_activity_prediction Predict the activity state of a ligand based on a classification model that was trained to predict ligand activity state based on ligand importance scores.

Usage

```
model_based_ligand_activity_prediction(importances, model, normalization)
```

Arguments

importances	A data frame containing at least folowing variables: $setting, $test_ligand, $ligand and one or more feature importance scores. $test_ligand denotes the name of a possibly active ligand, $ligand the name of the truely active ligand.
model	A model object of a classification object as e.g. generated via caret.
normalization	Way of normalization of the importance measures: "mean" (classifcal z-score) or "median" (modified z-score)

Value

A data frame containing the ligand importance scores and the probabilities that according to the trained model, the ligands are active based on their importance scores.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
evaluation <- evaluate_importances_ligand_prediction(ligand_importances, "median", "lda")

settings <- lapply(expression_settings_validation[5:10], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
activity_predictions <- model_based_ligand_activity_prediction(ligand_importances, evaluation$model, "median")
print(head(activity_predictions))

## End(Not run)
```

---

model_evaluation_hyperparameter_optimization_mlrmbo

Construct and evaluate a ligand-target model given input parameters with the purpose of hyperparameter optimization using mlrMBO.

Description

model_evaluation_hyperparameter_optimization_mlrmbo will take as input a setting of parameters (hyperparameters), data source weights and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input validation settings (average performance for both target gene prediction and ligand activity prediction, as measured via the auroc and aupr).

Usage

```
model_evaluation_hyperparameter_optimization_mlrmbo(x, source_weights, algorithm, correct_topology, l
```

Arguments

x	A list containing the following elements. $lr_sig_hub: hub correction factor for the ligand-signaling network; $gr_hub: hub correction factor for the gene regulatory network; $damping_factor: damping factor in the PPR algorithm if using PPR and optionally $ltf_cutoff: the cutoff on the ligand-tf matrix. For more information about these parameters: see `construct_ligand_target_matrix` and `apply_hub_correction`.
source_weights	A named numeric vector indicating the weight for every data source.
algorithm	Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights). Default and recommended: PPR.
correct_topology	This parameter indicates whether the PPR-constructed ligand-target matrix will be subtracted by a PR-constructed target matrix. TRUE or FALSE.
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
settings	A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
secondary_targets	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE
remove_direct_links	Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"
damping_factor	The value of the damping factor if damping factor is a fixed parameter and will not be optimized and thus not belong to x. Default NULL.
...	Additional arguments to `make_discrete_ligand_target_matrix`.

Value

A numeric vector of length 4 containing the average auroc for target gene prediction, average aupr (corrected for TP fraction) for target gene prediction, average auroc for ligand activity prediction and average aupr for ligand activity prediction.

Examples

```
## Not run:
library(dplyr)
nr_datasources <- source_weights_df$source %>%
  unique() %>%
  length()
test_input <- list("lr_sig_hub" = 0.5, "gr_hub" = 0.5, "damping_factor" = 0.5)
source_weights <- source_weights_df$weight
names(source_weights) <- source_weights_df$source

## End(Not run)
```

`model_evaluation_optimization_application`

Construct and evaluate a ligand-target model given input parameters with the purpose of parameter optimization for multi-ligand application.

Description

`model_evaluation_optimization_application` will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input application settings (average performance for target gene prediction, as measured via the auroc and aupr).

Usage

```
model_evaluation_optimization_application(x, source_names, algorithm, correct_topology, lr_network, s
```

Arguments

`x`	A list containing parameter values for parameter optimization. $source_weights: numeric vector representing the weight for each data source; $lr_sig_hub: hub correction factor for the ligand-signaling network; $gr_hub: hub correction factor for the gene regulatory network; $damping_factor: damping factor in the PPR algorithm if using PPR and optionally $ltf_cutoff: the cutoff on the ligand-tf matrix. For more information about these parameters: see `construct_ligand_target_matrix` and `apply_hub_correction`.
`source_names`	Character vector containing the names of the data sources. The order of data source names accords to the order of weights in x$source_weights.

`algorithm` Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights). Default and recommended: PPR.

`correct_topology` This parameter indicates whether the PPR-constructed ligand-target matrix will be subtracted by a PR-constructed target matrix. TRUE or FALSE.

`lr_network` A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)

`sig_network` A data frame / tibble containing signaling interactions (required columns: from, to, source)

`gr_network` A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)

`settings` A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.

`secondary_targets` Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

`remove_direct_links` Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

`classification_algorithm` The name of the classification algorithm to be applied. Should be supported by the caret package. Examples of algorithms we recommend: with embedded feature selection: "rf","glm","fda","glmnet","sdwd","gam","glmboost", "pls" (load "pls" package before!); without: "lda","naive_bayes", "pcaNNet". Please notice that not all these algorithms work when the features (i.e. ligand vectors) are categorical (i.e. discrete class assignments).

`damping_factor` The value of the damping factor if damping factor is a fixed parameter and will not be optimized and thus not belong to x. Default NULL.

`...` Additional arguments to `evaluate_multi_ligand_target_prediction`.

Value

A numeric vector of length 2 containing the average auroc and aupr for target gene prediction.

Examples

```
## Not run:
library(dplyr)
nr_datasources <- source_weights_df$source %>%
```

```
  unique() %>%
  length()
test_input <- list("source_weights" = rep(0.5, times = nr_datasources), "lr_sig_hub" = 0.5, "gr_hub" = 0.5, "dampin

## End(Not run)
```

`model_evaluation_optimization_mlrmbo` *Construct and evaluate a ligand-target model given input parameters with the purpose of parameter optimization with mlrMBO.*

Description

`model_evaluation_optimization_mlrmbo` will take as input a setting of parameters (data source weights and hyperparameters) and layer-specific networks to construct a ligand-target matrix and evaluate its performance on input validation settings (average performance for both target gene prediction and ligand activity prediction, as measured via the auroc and aupr).

Usage

```
model_evaluation_optimization_mlrmbo(x, source_names, algorithm, correct_topology, lr_network, sig_ne
```

Arguments

`x`	A list containing parameter values for parameter optimization. $source_weights: numeric vector representing the weight for each data source; $lr_sig_hub: hub correction factor for the ligand-signaling network; $gr_hub: hub correction factor for the gene regulatory network; $damping_factor: damping factor in the PPR algorithm if using PPR and optionally $ltf_cutoff: the cutoff on the ligand-tf matrix. For more information about these parameters: see `construct_ligand_target_matrix` and `apply_hub_correction`.
`source_names`	Character vector containing the names of the data sources. The order of data source names accords to the order of weights in x$source_weights.
`algorithm`	Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights). Default and recommended: PPR.
`correct_topology`	This parameter indicates whether the PPR-constructed ligand-target matrix will be subtracted by a PR-constructed target matrix. TRUE or FALSE.
`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)

gr_network — A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)

settings — A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.

secondary_targets — Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE

remove_direct_links — Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"

damping_factor — The value of the damping factor if damping factor is a fixed parameter and will not be optimized and thus not belong to x. Default NULL.

... — Additional arguments to make_discrete_ligand_target_matrix.

Value

A numeric vector of length 4 containing the average auroc for target gene prediction, average aupr (corrected for TP fraction) for target gene prediction, average auroc for ligand activity prediction and average aupr for ligand activity prediction.

Examples

```
## Not run:
library(dplyr)
nr_datasources <- source_weights_df$source %>%
  unique() %>%
  length()
test_input <- list("source_weights" = rep(0.5, times = nr_datasources), "lr_sig_hub" = 0.5, "gr_hub" = 0.5, "dampin
expression_settings_validation <- readRDS(url("https://zenodo.org/record/3260758/files/expression_settings.rds"
# test_evaluation_optimization = model_evaluation_optimization_mlrmbo(test_input, source_weights_df$source %>% u

## End(Not run)
```

---

model_evaluation_optimization_nsga2r

Construct and evaluate a ligand-target model with the purpose of parameter optimization with NSGA-II.

Description

model_evaluation_optimization_nsga2 will take as input a vector of data source weights and hyperparameters to construct a ligand-target matrix and evaluate its performance on input validation settings.

Usage

```
model_evaluation_optimization_nsga2(y, source_names, algorithm, correct_topology, lr_network, sig_net
```

Arguments

y	A numeric vector containing the data source weights as the first elements, and hyperparameters as the last elements. The order of the data source weights accords to the order of source_names.
source_names	Character vector containing the names of the data sources. The order of data source names accords to the order of weights in x$source_weights.
algorithm	Selection of the algorithm to calculate ligand-tf signaling probability scores. Different options: "PPR" (personalized pagerank), "SPL" (shortest path length) and "direct"(just take weights of ligand-signaling network as ligand-tf weights). Default and recommended: PPR.
correct_topology	This parameter indicates whether the PPR-constructed ligand-target matrix will be subtracted by a PR-constructed target matrix. TRUE or FALSE.
lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
settings	A list of lists for which each sub-list contains the following elements: .$name: name of the setting; .$from: name(s) of the ligand(s) active in the setting of interest; .$response: named logical vector indicating whether a target is a TRUE target of the possibly active ligand(s) or a FALSE.
secondary_targets	Indicate whether a ligand-target matrix should be returned that explicitly includes putative secondary targets of a ligand (by means of an additional matrix multiplication step considering primary targets as possible regulators). Default: FALSE
remove_direct_links	Indicate whether direct ligand-target and receptor-target links in the gene regulatory network should be kept or not. "no": keep links; "ligand": remove direct ligand-target links; "ligand-receptor": remove both direct ligand-target and receptor-target links. Default: "no"
damping_factor	The value of the damping factor if damping factor is a fixed parameter and will not be optimized and thus not belong to x. Default NULL.

Value

A numeric vector of length 4 containing the average auroc for target gene prediction, average aupr (corrected for TP fraction) for target gene prediction, average auroc for ligand activity prediction and average aupr for ligand activity prediction.

Examples

```
## Not run:
nr_datasources <- source_weights_df$source %>%
  unique() %>%
  length()
test_input <- c(rep(0.5, times = nr_datasources), 0.5, 0.5, 0.5, 0.5)
expression_settings_validation <- readRDS(url("https://zenodo.org/record/3260758/files/expression_settings.rds"
test_evaluation_optimization <- model_evaluation_optimization_nsga2(test_input, source_weights_df$source %>% uni
  lapply(expression_settings_validation, convert_expression_settings_evaluation),
  secondary_targets = FALSE, remove_direct_links = "no"
)

## End(Not run)
```

---

mutate_cond *Change values in a tibble if some condition is fulfilled.*

Description

mutate_cond Change values in a tibble if some condition is fulfilled. Credits: https://stackoverflow.com/questions/34096162/
mutate-replace-several-columns-on-a-subset-of-rows.

Usage

```
mutate_cond(.data, condition, ..., envir = parent.frame())
```

Arguments

.data	Data frame / tibble
condition	A condition that need to be fulfilled.
...	The change that need to happen if condition fulfilled – through the use of 'dplyr::mutate()'
envir	parent.frame() by default

Value

A tibble

Examples

```
## Not run:
mutate_cond(df, a == 3, b = 4)

## End(Not run)
```

`ncitations` *Number of citations for genes*

Description

A data.frame/tibble describing the number of times a particular gene is cited in the pubmed literature (date: 13-03-2018)

Usage

```
ncitations
```

Format

A data frame/tibble

entrez entrez

ncitations ncitations

symbol symbol

entrez_mouse entrez_mouse

symbol_mouse symbol_mouse

`nichenet_seuratobj_aggregate` *Perform NicheNet analysis on Seurat object: explain DE between conditions*

Description

`nichenet_seuratobj_aggregate` Perform NicheNet analysis on Seurat object: explain differential expression (DE) in a receiver celltype between two different conditions by ligands expressed by sender cells

Usage

```
nichenet_seuratobj_aggregate(
  receiver,
  seurat_obj,
  condition_colname,
  condition_oi,
  condition_reference,
  sender = "all",
  ligand_target_matrix,
  lr_network,
  weighted_networks,
  assay_oi = NULL,
  expression_pct = 0.1,
  lfc_cutoff = 0.25,
  geneset = "DE",
  filter_top_ligands = TRUE,
  top_n_ligands = 30,
  top_n_targets = 200,
  cutoff_visualization = 0.33,
  verbose = TRUE
)
```

Arguments

receiver	Name of cluster identity/identities of cells that are presumably affected by intercellular communication with other cells
seurat_obj	Single-cell expression dataset as Seurat object https://satijalab.org/seurat/.
condition_colname	Name of the column in the meta data dataframe that indicates which condition/sample cells were coming from.
condition_oi	Condition of interest in which receiver cells were presumably affected by other cells. Should be a name present in the 'condition_colname' column of the metadata.
condition_reference	The second condition (e.g. reference or steady-state condition). Should be a name present in the 'condition_colname' column of the metadata.
sender	Determine the potential sender cells. Name of cluster identity/identities of cells that presumably affect expression in the receiver cell type. In case you want to look at all possible sender cell types in the data, you can give this argument the value "all". "all" indicates thus that all cell types in the dataset will be considered as possible sender cells. As final option, you could give this argument the value "undefined"."undefined" won't look at ligands expressed by sender cells, but at all ligands for which a corresponding receptor is expressed. This could be useful if the presumably active sender cell is not profiled. Default: "all".
ligand_target_matrix	The NicheNet ligand-target matrix of the organism of interest denoting regulatory potential scores between ligands and targets (ligands in columns).

`lr_network` The ligand-receptor network (columns that should be present: $from, $to) of the organism of interest.

`weighted_networks` The NicheNet weighted networks of the organism of interest denoting interactions and their weights/confidences in the ligand-signaling and gene regulatory network.

`assay_oi` The assay to be used for calculating expressed genes and the DE analysis. If NULL, the default assay of the Seurat object will be used.

`expression_pct` To determine ligands and receptors expressed by sender and receiver cells, we consider genes expressed if they are expressed in at least a specific fraction of cells of a cluster. This number indicates this fraction. Default: 0.10

`lfc_cutoff` Cutoff on log fold change in the wilcoxon differential expression test. Default: 0.25.

`geneset` Indicate whether to consider all DE genes between condition 1 and 2 ("DE"), or only genes upregulated in condition 1 ("up"), or only genes downregulad in condition 1 ("down").

`filter_top_ligands` Indicate whether output tables for ligand-target and ligand-receptor networks should be done for a filtered set of top ligands (TRUE) or for all ligands (FALSE). Default: TRUE.

`top_n_ligands` Indicate how many ligands should be extracted as top-ligands after ligand activity analysis. Only for these ligands, target genes and receptors will be returned. Default: 30.

`top_n_targets` To predict active, affected targets of the prioritized ligands, consider only DE genes if they also belong to the a priori top n ("top_n_targets") targets of a ligand. Default = 200.

`cutoff_visualization` Because almost no ligand-target scores have a regulatory potential score of 0, we clarify the heatmap visualization by giving the links with the lowest scores a score of 0. The cutoff_visualization paramter indicates this fraction of links that are given a score of zero. Default = 0.33.

`verbose` Print out the current analysis stage. Default: TRUE.

Value

A list with the following elements: $ligand_activities: data frame with output ligand activity analysis; $top_ligands: top_n ligands based on ligand activity; $top_targets: active, affected target genes of these ligands; $top_receptors: receptors of these ligands; $ligand_target_matrix: matrix indicating regulatory potential scores between active ligands and their predicted targets; $ligand_target_heatmap: heatmap of ligand-target regulatory potential; $ligand_target_df: data frame showing regulatory potential scores of predicted active ligand-target network; $ligand_activity_target_heatmap: heatmap showing both ligand activity scores and target genes of these top ligands; $ligand_expression_dotplot: expression dotplot of the top ligands; $ligand_differential_expression_heatmap = differential expression heatmap of the top ligands; $ligand_receptor_matrix: matrix of ligand-receptor interactions; $ligand_receptor_heatmap: heatmap showing ligand-receptor interactions; $ligand_receptor_df: data frame of ligand-receptor interactions; $geneset_oi: a vector containing the set of genes used as

input for the ligand activity analysis; $background_expressed_genes: the background of genes to which the geneset will be compared in the ligand activity analysis.

Examples

```
## Not run:
seuratObj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj_test.rds"))
lr_network <- readRDS(url("https://zenodo.org/record/7074291/files/lr_network_mouse_21122021.rds"))
ligand_target_matrix <- readRDS(url("https://zenodo.org/record/7074291/files/ligand_target_matrix_nsga2r_final_
weighted_networks <- readRDS(url("https://zenodo.org/record/7074291/files/weighted_networks_nsga2r_final_mouse.
nichenet_seuratobj_aggregate(receiver = "CD8 T", seurat_obj = seuratObj, condition_colname = "aggregate", conditio

## End(Not run)
```

---

nichenet_seuratobj_aggregate_cluster_de

Perform NicheNet analysis on Seurat object: explain DE between two cell clusters from separate conditions

---

Description

nichenet_seuratobj_aggregate_cluster_de Perform NicheNet analysis on Seurat object: explain differential expression (DE) between two 'receiver' cell clusters coming from different conditions, by ligands expressed by neighboring cells.

Usage

```
nichenet_seuratobj_aggregate_cluster_de(seurat_obj, receiver_affected, receiver_reference, condition
```

Arguments

seurat_obj	Single-cell expression dataset as Seurat object https://satijalab.org/seurat/.
receiver_affected	Name of cluster identity/identities of "affected" cells that were presumably affected by intercellular communication with other cells
receiver_reference	Name of cluster identity/identities of "steady-state" cells, before they are affected by intercellular communication with other cells
condition_colname	Name of the column in the meta data dataframe that indicates which condition/sample cells were coming from.
condition_oi	Condition of interest in which receiver cells were presumably affected by other cells. Should be a name present in the 'condition_colname' column of the metadata.
condition_reference	The second condition (e.g. reference or steady-state condition). Should be a name present in the 'condition_colname' column of the metadata.

`sender` Determine the potential sender cells. Name of cluster identity/identities of cells that presumably affect expression in the receiver cell type. In case you want to look at all possible sender cell types in the data, you can give this argument the value "all". "all" indicates thus that all cell types in the dataset will be considered as possible sender cells. As final option, you could give this argument the value "undefined"."undefined" won't look at ligands expressed by sender cells, but at all ligands for which a corresponding receptor is expressed. This could be useful if the presumably active sender cell is not profiled. Default: "all".

`ligand_target_matrix` The NicheNet ligand-target matrix of the organism of interest denoting regulatory potential scores between ligands and targets (ligands in columns).

`lr_network` The ligand-receptor network (columns that should be present: $from, $to) of the organism of interest.

`weighted_networks` The NicheNet weighted networks of the organism of interest denoting interactions and their weights/confidences in the ligand-signaling and gene regulatory network.

`assay_oi` The assay to be used for calculating expressed genes and the DE analysis. If NULL, the default assay of the Seurat object will be used.

`expression_pct` To determine ligands and receptors expressed by sender and receiver cells, we consider genes expressed if they are expressed in at least a specific fraction of cells of a cluster. This number indicates this fraction. Default: 0.10

`lfc_cutoff` Cutoff on log fold change in the wilcoxon differential expression test. Default: 0.25.

`geneset` Indicate whether to consider all DE genes between condition 1 and 2 ("DE"), or only genes upregulated in condition 1 ("up"), or only genes downregulad in condition 1 ("down").

`filter_top_ligands` Indicate whether output tables for ligand-target and ligand-receptor networks should be done for a filtered set of top ligands (TRUE) or for all ligands (FALSE). Default: TRUE.

`top_n_ligands` Indicate how many ligands should be extracted as top-ligands after ligand activity analysis. Only for these ligands, target genes and receptors will be returned. Default: 30.

`top_n_targets` To predict active, affected targets of the prioritized ligands, consider only DE genes if they also belong to the a priori top n ("top_n_targets") targets of a ligand. Default = 200.

`cutoff_visualization` Because almost no ligand-target scores have a regulatory potential score of 0, we clarify the heatmap visualization by giving the links with the lowest scores a score of 0. The cutoff_visualization paramter indicates this fraction of links that are given a score of zero. Default = 0.33.

`verbose` Print out the current analysis stage. Default: TRUE.

Value

A list with the following elements: $ligand_activities: data frame with output ligand activity analysis; $top_ligands: top_n ligands based on ligand activity; $top_targets: active, affected target genes of these ligands; $top_receptors: receptors of these ligands; $ligand_target_matrix: matrix indicating regulatory potential scores between active ligands and their predicted targets; $ligand_target_heatmap: heatmap of ligand-target regulatory potential; $ligand_target_df: data frame showing regulatory potential scores of predicted active ligand-target network; $ligand_activity_target_heatmap: heatmap showing both ligand activity scores and target genes of these top ligands; $ligand_expression_dotplot: expression dotplot of the top ligands; $ligand_receptor_matrix: matrix of ligand-receptor interactions; $ligand_receptor_heatmap: heatmap showing ligand-receptor interactions; $ligand_receptor_df: data frame of ligand-receptor interactions; $geneset_oi: a vector containing the set of genes used as input for the ligand activity analysis; $background_expressed_genes: the background of genes to which the geneset will be compared in the ligand activity analysis.

Examples

```
## Not run:
seuratObj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj_test.rds"))
lr_network <- readRDS(url("https://zenodo.org/record/7074291/files/lr_network_mouse_21122021.rds"))
ligand_target_matrix <- readRDS(url("https://zenodo.org/record/7074291/files/ligand_target_matrix_nsga2r_final_
weighted_networks <- readRDS(url("https://zenodo.org/record/7074291/files/weighted_networks_nsga2r_final_mouse.
nichenet_seuratobj_aggregate_cluster_de(seurat_obj = seuratObj, receiver_affected = "CD8 T", receiver_reference =

## End(Not run)
```

---

nichenet_seuratobj_cluster_de

Perform NicheNet analysis on Seurat object: explain DE between two cell clusters

---

Description

nichenet_seuratobj_cluster_de Perform NicheNet analysis on Seurat object: explain differential expression (DE) between two 'receiver' cell clusters by ligands expressed by neighboring cells.

Usage

```
nichenet_seuratobj_cluster_de(seurat_obj, receiver_affected, receiver_reference, sender = "all",ligan
```

Arguments

seurat_obj	Single-cell expression dataset as Seurat object https://satijalab.org/seurat/.
receiver_affected	Name of cluster identity/identities of "affected" cells that were presumably affected by intercellular communication with other cells

receiver_reference
: Name of cluster identity/identities of "steady-state" cells, before they are affected by intercellular communication with other cells

sender
: Determine the potential sender cells. Name of cluster identity/identities of cells that presumably affect expression in the receiver cell type. In case you want to look at all possible sender cell types in the data, you can give this argument the value "all". "all" indicates thus that all cell types in the dataset will be considered as possible sender cells. As final option, you could give this argument the value "undefined"."undefined" won't look at ligands expressed by sender cells, but at all ligands for which a corresponding receptor is expressed. This could be useful if the presumably active sender cell is not profiled. Default: "all".

ligand_target_matrix
: The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).

lr_network
: The ligand-receptor network (columns that should be present: $from, $to).

weighted_networks
: The NicheNet weighted networks denoting interactions and their weights/confidences in the ligand-signaling and gene regulatory network.

assay_oi
: The assay to be used for calculating expressed genes and the DE analysis. If NULL, the default assay of the Seurat object will be used.

expression_pct
: To determine ligands and receptors expressed by sender and receiver cells, we consider genes expressed if they are expressed in at least a specific fraction of cells of a cluster. This number indicates this fraction. Default: 0.10

lfc_cutoff
: Cutoff on log fold change in the wilcoxon differential expression test. Default: 0.25.

geneset
: Indicate whether to consider all DE genes between condition 1 and 2 ("DE"), or only genes upregulated in condition 1 ("up"), or only genes downregulad in condition 1 ("down").

filter_top_ligands
: Indicate whether output tables for ligand-target and ligand-receptor networks should be done for a filtered set of top ligands (TRUE) or for all ligands (FALSE). Default: TRUE.

top_n_ligands
: Indicate how many ligands should be extracted as top-ligands after ligand activity analysis. Only for these ligands, target genes and receptors will be returned. Default: 30.

top_n_targets
: To predict active, affected targets of the prioritized ligands, consider only DE genes if they also belong to the a priori top n ("top_n_targets") targets of a ligand. Default = 200.

cutoff_visualization
: Because almost no ligand-target scores have a regulatory potential score of 0, we clarify the heatmap visualization by giving the links with the lowest scores a score of 0. The cutoff_visualization paramter indicates this fraction of links that are given a score of zero. Default = 0.33.

verbose
: Print out the current analysis stage. Default: TRUE.

Value

A list with the following elements: $ligand_activities: data frame with output ligand activity analysis; $top_ligands: top_n ligands based on ligand activity; $top_targets: active, affected target genes of these ligands; $top_receptors: receptors of these ligands; $ligand_target_matrix: matrix indicating regulatory potential scores between active ligands and their predicted targets; $ligand_target_heatmap: heatmap of ligand-target regulatory potential; $ligand_target_df: data frame showing regulatory potential scores of predicted active ligand-target network; $ligand_activity_target_heatmap: heatmap showing both ligand activity scores and target genes of these top ligands; $ligand_expression_dotplot: expression dotplot of the top ligands; $ligand_receptor_matrix: matrix of ligand-receptor interactions; $ligand_receptor_heatmap: heatmap showing ligand-receptor interactions; $ligand_receptor_df: data frame of ligand-receptor interactions; $geneset_oi: a vector containing the set of genes used as input for the ligand activity analysis; $background_expressed_genes: the background of genes to which the geneset will be compared in the ligand activity analysis.

Examples

```
## Not run:
seuratObj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj_test.rds"))
lr_network <- readRDS(url("https://zenodo.org/record/7074291/files/lr_network_mouse_21122021.rds"))
ligand_target_matrix <- readRDS(url("https://zenodo.org/record/7074291/files/ligand_target_matrix_nsga2r_final_
weighted_networks <- readRDS(url("https://zenodo.org/record/7074291/files/weighted_networks_nsga2r_final_mouse.
# works, but does not make sense
nichenet_seuratobj_cluster_de(seurat_obj = seuratObj, receiver_affected = "CD8 T", receiver_reference = "Mono", se
# type of analysis for which this would make sense
nichenet_seuratobj_cluster_de(seurat_obj = seuratObj, receiver_affected = "p-EMT-pos-cancer", receiver_reference

## End(Not run)
```

---

normalize_single_cell_ligand_activities

Normalize single-cell ligand activities

Description

normalize_single_cell_ligand_activities Normalize single-cell ligand activities to make ligand activities over different cells comparable.

Usage

```
normalize_single_cell_ligand_activities(ligand_activities)
```

Arguments

ligand_activities	Output from the function 'predict_single_cell_ligand_activities'.

Value

A tibble giving the normalized ligand activity scores for single cells. Following columns in the tibble: $cell, $ligand, $pearson, which is the normalized ligand activity value.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
cell_ids <- c("cell1", "cell2")
expression_scaled <- matrix(rnorm(length(genes) * 2, sd = 0.5, mean = 0.5), nrow = 2)
rownames(expression_scaled) <- cell_ids
colnames(expression_scaled) <- genes
ligand_activities <- predict_single_cell_ligand_activities(cell_ids = cell_ids, expression_scaled = expression_sc
normalized_ligand_activities <- normalize_single_cell_ligand_activities(ligand_activities)

## End(Not run)
```

---

optimized_source_weights_df

Optimized data source weights

Description

A data.frame/tibble describing weight associated to each data source. These weights will be used for weighted aggregation of source networks and were determined via parameter optimization (see NicheNet paper).

Usage

```
optimized_source_weights_df
```

Format

A data frame/tibble

source name of data source

avg_weight weight associated to a data source, averaged over cross-validation folds

median_weight weight associated to a data source, averaged over cross-validation folds

predict_ligand_activities *Predict activities of ligands in regulating expression of a gene set of interest*

Description

predict_ligand_activities Predict activities of ligands in regulating expression of a gene set of interest. Ligand activities are defined as how well they predict the observed transcriptional response (i.e. gene set) according to the NicheNet model.

Usage

```
predict_ligand_activities(geneset, background_expressed_genes,ligand_target_matrix, potential_ligand
```

Arguments

geneset	Character vector of the gene symbols of genes of which the expression is potentially affected by ligands from the interacting cell.
background_expressed_genes	Character vector of gene symbols of the background, non-affected, genes (can contain the symbols of the affected genes as well).
ligand_target_matrix	The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).
potential_ligands	Character vector giving the gene symbols of the potentially active ligands you want to define ligand activities for.
single	TRUE if you want to calculate ligand activity scores by considering every ligand individually (recommended). FALSE if you want to calculate ligand activity scores as variable importances of a multi-ligand classification model.
...	Additional parameters for get_multi_ligand_importances if single = FALSE.

Value

A tibble giving several ligand activity scores. Following columns in the tibble: $test_ligand, $auroc, $aupr and $pearson.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
geneset <- c("SOCS2", "SOCS3", "IRF1")
background_expressed_genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
```

```
ligand_activities <- predict_ligand_activities(geneset = geneset, background_expressed_genes = background_express

## End(Not run)
```

`predict_single_cell_ligand_activities`
Single-cell ligand activity prediction

Description

`predict_single_cell_ligand_activities` For every individual cell of interest, predict activities of ligands in regulating expression of genes that are stronger expressed in that cell compared to other cells (0.975 quantile). Ligand activities are defined as how well they predict the observed transcriptional response (i.e. gene set) according to the NicheNet model.

Usage

```
predict_single_cell_ligand_activities(cell_ids, expression_scaled,ligand_target_matrix, potential_li
```

Arguments

`cell_ids`	Identities of cells for which the ligand activities should be calculated.
`expression_scaled`	Scaled expression matrix of single-cells (scaled such that high values indicate that a gene is stronger expressed in that cell compared to others)
`ligand_target_matrix`	The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).
`potential_ligands`	Character vector giving the gene symbols of the potentially active ligands you want to define ligand activities for.
`single`	TRUE if you want to calculate ligand activity scores by considering every ligand individually (recommended). FALSE if you want to calculate ligand activity scores as variable importances of a multi-ligand classification model.
`...`	Additional parameters for get_multi_ligand_importances if single = FALSE.

Value

A tibble giving several ligand activity scores for single cells. Following columns in the tibble: $setting, $test_ligand, $auroc, $aupr and $pearson.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
cell_ids <- c("cell1", "cell2")
expression_scaled <- matrix(rnorm(length(genes) * 2, sd = 0.5, mean = 0.5), nrow = 2)
rownames(expression_scaled) <- cell_ids
colnames(expression_scaled) <- genes
ligand_activities <- predict_single_cell_ligand_activities(cell_ids = cell_ids, expression_scaled = expression_sc

## End(Not run)
```

---

prepare_circos_visualization

Prepare circos visualization

Description

Prepare the data for the circos visualization by incorporating the colors and order of the links, as well as gaps between different cell types

Usage

```
prepare_circos_visualization(circos_links, ligand_colors = NULL, target_colors = NULL, widths = NULL, c
```

Arguments

circos_links	Dataframe from the function get_ligand_target_links_oi containing weighted ligand-target links, cell type expressing the ligand, and target gene goruping
ligand_colors	Named vector of colors for each cell type (default: NULL, where colors follow the ggplot default color scheme)
target_colors	Named vector of colors for each target gene grouping (default: NULL, where colors follow the ggplot default color scheme)
widths	Named list of widths for the different types groupings, including: • width_same_cell_same_ligand_type: Width of the links between ligands of the same cell type (default: 0.5) • width_different_cell: Width of the links between different cell types, or between different target gene groups (default: 6) • width_ligand_target: Width of the links between ligands and targets (default: 15) • width_same_cell_same_target_type: Width of the links between target genes of the same group (default: 0.5)

celltype_order Order of the cell types (default: NULL, where cell types are ordered alphabetically, followed by "General"). Cell types are then drawn counter-clockwise in the circos plot.

Value

A list of four objects, including:

- circos_links: Dataframe of weighted ligand-target links
- ligand_colors: Named vector of ligands and their colors
- order: Vector of order of the ligands and target genes
- gaps: Vector of gaps between the different groupings

Examples

```
## Not run:
celltype_order <- c("General", "NK", "B", "DC", "Mono")
ligand_colors <- c("General" = "lawngreen", "NK" = "royalblue", "B" = "darkgreen", "Mono" = "violet", "DC" = "steelb
target_colors <- c("LCMV-DE" = "tomato")
vis_circos_obj <- prepare_circos_visualization(circos_links, ligand_colors, target_colors, celltype_order = cellt

## End(Not run)
```

---

prepare_ligand_receptor_visualization

Prepare ligand-receptor visualization

Description

prepare_ligand_receptor_visualization Prepare a matrix of ligand-receptor interactions for visualization.

Usage

```
prepare_ligand_receptor_visualization(
  lr_network_top_df_long,
  best_upstream_ligands,
  order_hclust = "both"
)
```

Arguments

lr_network_top_df_long
A data frame with three columns, from, to and weight, containing the ligand-receptor interactions and their weights.

best_upstream_ligands
: Character vector of ligands of interest. This will only be used if order_hclust = "receptors" or order_hclust = "none".

order_hclust
: "both", "ligands", "receptors", or "none". If "both", the ligands and receptors are ordered by hierarchical clustering. If "ligands" or "receptors" only the ligands or receptors are ordered hierarchically. If "none", no hierarchical clustering is performed, and the ligands are ordered based on best_upstream_ligands, and the receptors are ordered alphabetically.

Value

A matrix of ligand-receptor interactions for visualization.

Examples

```
## Not run:
ligand_receptor_network <- prepare_ligand_receptor_visualization(best_upstream_ligands = best_upstream_ligands,

## End(Not run)
```

---

prepare_ligand_target_visualization

Prepare heatmap visualization of the ligand-target links starting from a ligand-target tibble.

Description

prepare_ligand_target_visualization Prepare heatmap visualization of the ligand-target links starting from a ligand-target tibble. Get regulatory potential scores between all pairs of ligands and targets documented in this tibble. For better visualization, we propose to define a quantile cutoff on the ligand-target scores.

Usage

```
prepare_ligand_target_visualization(ligand_target_df, ligand_target_matrix, cutoff = 0.25)
```

Arguments

ligand_target_df
: Tibble with columns 'ligand', 'target' and 'weight' to indicate ligand-target regulatory potential scores of interest.

ligand_target_matrix
: The NicheNet ligand-target matrix denoting regulatory potential scores between ligands and targets (ligands in columns).

cutoff
: Quantile cutoff on the ligand-target scores of the input weighted ligand-target network. Scores under this cutoff will be set to 0.

Value

A matrix giving the ligand-target regulatory potential scores between ligands of interest and their targets genes part of the gene set of interest.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
geneset <- c("SOCS2", "SOCS3", "IRF1")
background_expressed_genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
active_ligand_target_links_df <- potential_ligands %>%
 lapply(get_weighted_ligand_target_links, geneset = geneset, ligand_target_matrix = ligand_target_matrix, n = 250
  bind_rows()
active_ligand_target_links <- prepare_ligand_target_visualization(ligand_target_df = active_ligand_target_links_

## End(Not run)
```

---

prepare_settings_leave_one_in_characterization

Prepare settings for leave-one-in characterization

Description

`prepare_settings_leave_one_in_characterization` will generate a list of lists containing the data source weights that need to be used for model construction. Every sub-list will contain the data source weights needed to make so called leave-one-in models in which only one ligand-signaling data source is used and all gene regulatory data sources (or vice versa).

Usage

```
prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, source_weights_d
```

Arguments

`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)
`gr_network`	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
`source_weights_df`	A data frame / tibble containing the weights associated to each individual data source. Sources with higher weights will contribute more to the final model performance (required columns: source, weight). Note that only interactions described by sources included here, will be retained during model construction.

Value

A list of lists. Every sub-list contains 2 elements: $model_name: the name of the left-in data source; $source_weights: named numeric vector containing the data source weights that will be used for the construction of leave-one-in models.

Examples

```
## Not run:
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour

## End(Not run)
```

---

prepare_settings_leave_one_out_characterization

Prepare settings for leave-one-out characterization

Description

`prepare_settings_leave_one_out_characterization` will generate a list of lists containing the data source weights that need to be used for model construction. Every sub-list will contain the data source weights needed to make so called leave-one-out models in one ligand-signaling data source is or gene regulatory data source is left out.

Usage

```
prepare_settings_leave_one_out_characterization(lr_network, sig_network, gr_network, source_weights_
```

Arguments

`lr_network`	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
`sig_network`	A data frame / tibble containing signaling interactions (required columns: from, to, source)
`gr_network`	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
`source_weights_df`	A data frame / tibble containing the weights associated to each individual data source. Sources with higher weights will contribute more to the final model performance (required columns: source, weight). Note that only interactions described by sources included here, will be retained during model construction.

Value

A list of lists. Every sub-list contains 2 elements: $model_name: the name of the left-out data source; $source_weights: named numeric vector containing the data source weights that will be used for the construction of leave-one-out models.

prepare_settings_one_vs_one_characterization
Prepare settings for one-vs-one characterization

Description

prepare_settings_one_vs_one_characterization will generate a list of lists containing the data source weights that need to be used for model construction. Every sub-list will contain the data source weights needed to make so called one-vs-one models in which only one ligand-signaling data source and only one gene regulatory data source is used. It is also possible to construct one-vs-one-vs-one models by keeping 1 ligand-receptor, 1 signaling and 1 gene regulatory data source.

Usage

```
prepare_settings_one_vs_one_characterization(lr_network, sig_network, gr_network, lr_network_separat
```

Arguments

lr_network	A data frame / tibble containing ligand-receptor interactions (required columns: from, to, source)
sig_network	A data frame / tibble containing signaling interactions (required columns: from, to, source)
gr_network	A data frame / tibble containing gene regulatory interactions (required columns: from, to, source)
lr_network_separate	Indicate whether the one-vs-one models should contain 1 ligand-receptor, 1 signaling and 1 gene regulatory network source (TRUE) or just 1 ligand-signaling combined and 1 gene regulatory source (FALSE). Default: FALSE.

Value

A list of lists. Every sub-list contains following elements: $model_name; $source_lr_sig: the name of the left-in ligand-signaling data source (or $source_lr and $source_sig if lr_network_separate is TRUE); $source_gr: the name of the left-in gene regulatory data source and $source_weights: named numeric vector containing the data source weights that will be used for the construction of one-vs-one models.

Examples

```
## Not run:
weights_settings_ovo <- prepare_settings_one_vs_one_characterization(lr_network, sig_network, gr_network)
weights_settings_ovo_lr_separate <- prepare_settings_one_vs_one_characterization(lr_network, sig_network, gr_net

## End(Not run)
```

process_characterization_ligand_prediction
Process the output of model evaluation for data source characterization purposes on the ligand prediction performance

Description

process_characterization_ligand_prediction will process output formed by model evaluation to get a data frame containing performance measures in ligand prediction.

Usage

```
process_characterization_ligand_prediction(output_characterization)
```

Arguments

output_characterization
: a list of lists containing the results of evaluation of different models (e.g. after execution of evaluate_model. Every sublist should contain at least the following elements: $model_name and $performances_target_prediction.

Value

A data frame containing the ligand activity prediction performance for all the models that were evaluated.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network
ligand_prediction_performances <- process_characterization_ligand_prediction(output_characterization)

## End(Not run)
```

process_characterization_ligand_prediction_single_measures
Process the output of model evaluation for data source characterization purposes on the ligand prediction performance (for every importance score individually)

Description

process_characterization_ligand_prediction_single_measures will process output formed by model evaluation to get a data frame containing performance measures in ligand prediction for every importance score individually.

Usage

```
process_characterization_ligand_prediction_single_measures(output_characterization)
```

Arguments

output_characterization
: a list of lists containing the results of evaluation of different models (e.g. after execution of evaluate_model. Every sublist should contain at least the following elements: $model_name and $performances_target_prediction.

Value

A data frame containing the ligand activity prediction performance of every ligand importance measure for all the models that were evaluated.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network
ligand_prediction_performances <- process_characterization_ligand_prediction_single_measures(output_characteriz

## End(Not run)
```

---

process_characterization_popularity_slopes_ligand_prediction

Process the output of model evaluation for data source characterization purposes on the popularity bias assessment of ligand activity performance

---

Description

process_characterization_popularity_slopes_ligand_prediction will process output formed by model evaluation to get a data frame containing popularity bias measures in performance of ligand activity prediction.

Usage

```
process_characterization_popularity_slopes_ligand_prediction(output_characterization)
```

Arguments

output_characterization
: a list of lists containing the results of evaluation of different models (e.g. after execution of evaluate_model. Every sublist should contain at least the following elements: $model_name and $performances_target_prediction.

Value

A data frame containing the popularity bias slopes in ligand activity prediction performances.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network
popularity_slopes_ligand_prediction_performances <- process_characterization_popularity_slopes_ligand_predictio

## End(Not run)
```

---

process_characterization_popularity_slopes_target_prediction

Process the output of model evaluation for data source characterization purposes on the popularity bias assessment of target prediction performance

---

Description

process_characterization_popularity_slopes_target_prediction will process output formed by model evaluation to get a data frame containing popularity bias measures in performance of target gene prediction.

Usage

```
process_characterization_popularity_slopes_target_prediction(output_characterization)
```

Arguments

output_characterization
: a list of lists containing the results of evaluation of different models (e.g. after execution of evaluate_model. Every sublist should contain at least the following elements: $model_name and $performances_target_prediction.

Value

A data frame containing the popularity bias slopes in target gene prediction performances on every validation dataset for all the models that were evaluated.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network
popularity_slopes_target_prediction_performances <- process_characterization_popularity_slopes_target_predictio

## End(Not run)
```

---

process_characterization_target_prediction

Process the output of model evaluation for data source characterization purposes on the target prediction performance

---

Description

process_characterization_target_prediction will process output formed by model evaluation to get a data frame containing performance measures in target gene prediction.

Usage

```
process_characterization_target_prediction(output_characterization)
```

Arguments

output_characterization	a list of lists containing the results of evaluation of different models (e.g. after execution of evaluate_model. Every sublist should contain at least the following elements: $model_name and $performances_target_prediction.

Value

A data frame containing the target gene prediction performances on every validation dataset for all the models that were evaluated.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network
target_prediction_performances <- process_characterization_target_prediction(output_characterization)

## End(Not run)
```

---

process_characterization_target_prediction_average

Process the output of model evaluation for data source characterization purposes on the target prediction performance (average)

Description

`output_characterization` will process output formed by model evaluation to get a data frame containing performance measures in target gene prediction (averaged over all validation datasets).

Usage

```
process_characterization_target_prediction_average(output_characterization)
```

Arguments

`output_characterization`
: a list of lists containing the results of evaluation of different models (e.g. after execution of `evaluate_model`. Every sublist should contain at least the following elements: $model_name and $performances_target_prediction.

Value

A data frame containing the target gene prediction performances averaged over all validation datasets for all the models that were evaluated.

Examples

```
## Not run:
library(dplyr)
settings <- lapply(expression_settings_validation, convert_expression_settings_evaluation)
weights_settings_loi <- prepare_settings_leave_one_in_characterization(lr_network, sig_network, gr_network, sour
weights_settings_loi <- lapply(weights_settings_loi, add_hyperparameters_parameter_settings, lr_sig_hub = 0.25, g
doMC::registerDoMC(cores = 8)
output_characterization <- parallel::mclapply(weights_settings_loi[1:3], evaluate_model, lr_network, sig_network
target_prediction_performances <- process_characterization_target_prediction_average(output_characterization)
```

```
## End(Not run)
```

---

process_mlrmbo_nichenet_optimization *Process the output of mlrmbo multi-objective optimization to extract optimal parameter values.*

Description

process_mlrmbo_nichenet_optimization will process the output of multi-objective mlrmbo optimization. As a result, a list containing the optimal parameter values for model construction will be returned.

Usage

```
process_mlrmbo_nichenet_optimization(optimization_results,source_names,parameter_set_index = NULL)
```

Arguments

optimization_results	A list generated as output from multi-objective optimization by mlrMBO. Should contain the elements $pareto.front, $pareto.set See mlrmbo_optimization.
source_names	Character vector containing the names of the data sources. The order of data source names accords to the order of weights in x$source_weights.
parameter_set_index	Number indicating which of the proposed solutions must be selected to extract optimal parameters. If NULL: the solution with the highest geometric mean will be selected. Default: NULL.

Value

A list containing the parameter values leading to maximal performance and thus with the following elements: $source_weight_df, $lr_sig_hub, $gr_hub, $ltf_cutoff, $damping_factor

Examples

```
## Not run:
library(dplyr)
library(mlrMBO)
library(parallelMap)
additional_arguments_topology_correction <- list(source_names = source_weights_df$source %>% unique(), algorithm
nr_datasources <- additional_arguments_topology_correction$source_names %>% length()

obj_fun_multi_topology_correction <- makeMultiObjectiveFunction(name = "nichenet_optimization", description = "da

mlrmbo_optimization_result <- lapply(1, mlrmbo_optimization, obj_fun = obj_fun_multi_topology_correction, niter =
```

```
optimized_parameters <- process_mlrmbo_nichenet_optimization(mlrmbo_optimization_result[[1]], additional_argume

## End(Not run)
```

---

process_niche_de *Process the DE output of 'calculate_niche_de'*

Description

process_niche_de Process the DE output of 'calculate_niche_de': define what the average and minimum logFC value is after comparing a celltype vs all the celltypes of the other niches.

Usage

```
process_niche_de(DE_table, niches, type, expression_pct)
```

Arguments

DE_table	Output of 'calculate_niche_de'
niches	a list of lists/niches giving the name, senders and receiver celltypes for each nice. Sender and receiver cell types should be part of Idents(seurat_obj).
type	For what type of cellype is the DE analysis: "sender" or "receiver"?
expression_pct	Percentage of cells of a cell type having a non-zero expression value for a gene such that a gene can be considered expressed by that cell type.

Value

A tibble containing processed DE information

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
```

```
DE_table <- calculate_niche_de(seurat_obj, niches, "sender")
process_niche_de(DE_table, niches, "sender", expression_pct = 0.10)

## End(Not run)
```

---

process_receiver_target_de *Processing differential expression output of the receiver cell types*

Description

process_receiver_target_de Processing differential expression output of the receiver cell types – used before ligand-activity and ligand-target inference.

Usage

```
process_receiver_target_de(DE_receiver_targets, niches, expression_pct, specificity_score = "min_lfc"
```

Arguments

DE_receiver_targets	Output of 'calculate_niche_de' with 'type = receiver'
niches	a list of lists/niches giving the name, senders and receiver celltypes for each nice. Sender and receiver cell types should be part of Idents(seurat_obj).
expression_pct	Percentage of cells of a cell type having a non-zero expression value for a gene such that a gene can be considered expressed by that cell type.
specificity_score	Defines which score will be used to prioritze ligand-receptor pairs and consider their differential expression. Default and recommended: "min_lfc". "min_lfc" looks at the minimal logFC of the ligand/receptor between the celltype of interest and all the other celltypes. Alternatives: "mean_lfc", "min_score", and "mean_score". Mean uses the average/mean instead of minimum. score is the product of the logFC and the ratio of fraction of expressing cells.

Value

A tibble containing the processed DE information of the receiver cell types – used before ligand-activity and ligand-target inference.

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
niches <- list(
  "KC_niche" = list(
    "sender" = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
    "receiver" = c("KCs")
```

```
  ),
  "MoMac2_niche" = list(
    "sender" = c("Cholangiocytes", "Fibroblast 2"),
    "receiver" = c("MoMac2")
  ),
  "MoMac1_niche" = list(
    "sender" = c("Capsule fibroblasts", "Mesothelial cells"),
    "receiver" = c("MoMac1")
  )
)
DE_receiver <- calculate_niche_de(seurat_obj, niches, "receiver")
expression_pct <- 0.10
DE_receiver_processed <- process_receiver_target_de(DE_receiver_targets = DE_receiver, niches = niches, expressio

## End(Not run)
```

`process_spatial_de` *Process the spatialDE output*

Description

`process_spatial_de` Process the spatialDE output

Usage

```
process_spatial_de(DE_table, type, lr_network, expression_pct, specificity_score = "lfc")
```

Arguments

`DE_table`	Output of 'calculate_spatial_DE'
`type`	For what type of cellype is the DE analysis: "sender" or "receiver"?
`lr_network`	Ligand-Receptor Network in tibble format: ligand, receptor as columns
`expression_pct`	Percentage of cells of a cell type having a non-zero expression value for a gene such that a gene can be considered expressed by that cell type.
`specificity_score`	Defines which score will be used to prioritze ligand-receptor pairs and consider their differential expression. Default and recommended: "min_lfc". "min_lfc" looks at the minimal logFC of the ligand/receptor between the celltype of interest and all the other celltypes. Alternatives: "mean_lfc", "min_score", and "mean_score". Mean uses the average/mean instead of minimum. score is the product of the logFC and the ratio of fraction of expressing cells.

Value

A tibble of processed spatial DE information

Examples

```
## Not run:
seurat_obj <- readRDS(url("https://zenodo.org/record/5840787/files/seurat_obj_subset_integrated_zonation.rds"))
spatial_info <- tibble(
  celltype_region_oi = c("LSECs_portal", "Hepatocytes_portal", "Stellate cells_portal"),
 celltype_other_region = c("LSECs_central", "Hepatocytes_central", "Stellate cells_central")
) %>%
  mutate(niche = "KC_niche", celltype_type = "sender")
DE_table <- calculate_spatial_DE(seurat_obj, spatial_info)
processed_spatialDE <- process_spatial_de(DE_table, type = "sender", lr_network, expression_pct = 0.10, specificit

## End(Not run)
```

---

process_table_to_ic *Process DE or expression information into intercellular communication focused information.*

Description

process_table_to_ic First, only keep information of ligands for senders_oi, and information of receptors for receivers_oi. Then, combine information for senders and receivers by linking ligands to receptors based on the prior knowledge ligand-receptor network.

Usage

```
process_table_to_ic(table_object, table_type = "expression", lr_network, senders_oi = NULL, receivers_
```

Arguments

table_object	Output of get_exprs_avg, calculate_de, or FindMarkers
table_type	"expression", "celltype_DE", or "group_DE": indicates whether the table contains expression, celltype markers, or condition-specific information
lr_network	Prior knowledge Ligand-Receptor network (columns: ligand, receptor)
senders_oi	Default NULL: all celltypes will be considered as senders. If you want to select specific senders of interest: you can add this here as character vector.
receivers_oi	Default NULL: all celltypes will be considered as receivers If you want to select specific receivers of interest: you can add this here as character vector.

Value

Dataframe combining sender and receiver information linked to each other through joining by the ligand-receptor network.

Examples

```
## Not run:
library(dplyr)
lr_network <- readRDS(url("https://zenodo.org/record/3260758/files/lr_network.rds"))
lr_network <- lr_network %>%
  dplyr::rename(ligand = from, receptor = to) %>%
  dplyr::distinct(ligand, receptor)
seurat_obj <- readRDS(url("https://zenodo.org/record/3531889/files/seuratObj.rds"))
seurat_obj$celltype <- make.names(seuratObj$celltype)
# Calculate LCMV-specific average expression
expression_info <- get_exprs_avg(seurat_obj, "celltype", condition_oi = "LCMV", condition_colname = "aggregate")
# Calculate LCMV-specific cell-type markers
DE_table <- calculate_de(seurat_obj, "celltype", condition_oi = "LCMV", condition_colname = "aggregate")
# Calculate LCMV-specific genes across cell types
condition_markers <- FindMarkers(
  object = seuratObj, ident.1 = "LCMV", ident.2 = "SS",
  group.by = "aggregate", min.pct = 0, logfc.threshold = 0
) %>% rownames_to_column("gene")
processed_expr_info <- process_table_to_ic(expression_info, table_type = "expression", lr_network)
processed_DE_table <- process_table_to_ic(DE_table,
  table_type = "celltype_DE", lr_network,
  senders_oi = c("CD4.T", "Treg", "Mono", "NK", "B", "DC"), receivers_oi = "CD8.T"
)
processed_condition_markers <- process_table_to_ic(condition_markers, table_type = "condition_DE", lr_network)

## End(Not run)
```

---

randomize_complete_network_source_specific

Randomize an integrated network by shuffling its source networks

---

Description

`randomize_complete_network_source_specific` Randomizes an integrated network of interest by edge swapping in the source-specific networks to preserve the degree of the nodes

Usage

```
randomize_complete_network_source_specific(network)
```

Arguments

`network`	A data frame / tibble containing gene-gene interactions (required columns: $from, $to, $source)

Value

A randomized network ($from, $to, $source).

Examples

```
## Not run:
random_lr <- randomize_complete_network_source_specific(lr_network)

## End(Not run)
```

`randomize_datasource_network` *Randomize a network of a particular data source.*

Description

`randomize_datasource_network` Randomizes a network of a data source of interest by edge swapping to preserve the degree of the nodes.

Usage

```
randomize_datasource_network(datasource, network)
```

Arguments

`datasource`	The name of the data source for which the interactions need to be shuffled.
`network`	A data frame / tibble containing gene-gene interactions (required columns: $from, $to, $source)

Value

A randomized network ($from, $to, $source)

Examples

```
## Not run:
datasource_lr <- lr_network$source[1]
lr_randomized_source <- randomize_datasource_network(datasource_lr, lr_network)

## End(Not run)
```

randomize_network *Randomize a network*

Description

randomize_network Randomizes a network of interest by edge swapping to preserve the degree of the nodes

Usage

```
randomize_network(network,output_weighted = FALSE)
```

Arguments

network	A data frame / tibble containing gene-gene interactions (required columns: $from, $to)
output_weighted	Indicate whether the output network should be made weighted by assigning a weight of 1 to each interaction.

Value

A randomized network ($from, $to; and $weight = 1 if output_weighted == TRUE).

Examples

```
## Not run:
random_lr <- randomize_network(lr_network)

## End(Not run)
```

run_nsga2R_cluster *Run NSGA-II for parameter optimization.*

Description

run_nsga2R_cluster runs the NSGA-II algorithm for parameter optimization and allows for parallelization. The core of this function is adapted from nsga2R::nsga2R.

Usage

```
run_nsga2R_cluster(run_id, fn, varNo, objDim, lowerBounds = rep(-Inf, varNo), upperBounds = rep(Inf, va
```

Arguments

fn	The function to be optimized, usually model_evaluation_optimization_nsga2.
varNo	The number of variables to be optimized, usually the number of data sources + 4 hyperparameters (lr_sig_hub, gr_hub, ltf_cutoff, damping_factor).
objDim	Number of objective functions
lowerBounds	A numeric vector containing the lower bounds for the variables to be optimized (default: -Inf)
upperBounds	A numeric vector containing the upper bounds for the variables to be optimized (default: Inf)
popSize	The population size (default: 100)
tourSize	The tournament size (default: 2)
generations	The number of generations (default: 20)
ncores	The number of cores to be used for parallelization (default: 24)
cprob	The crossover probability (default: 0.7)
XoverDistIdx	The crossover distribution index, it can be any nonnegative real number (default: 5)
mprob	The mutation probability (default: 0.2)
MuDistIdx	The mutation distribution index, it can be any nonnegative real number (default: 10)
...	Additional arguments to fn.

Value

An 'nsga2R' object containing input settings and the following elements:

- intermed_output_list_params: a list with intermediate values of parameters for each generation (each element has dimensions popSize x varNo)
- intermed_output_list_obj: a list with intermediate values of objective functions for each generation (each element has dimensions popSize x objDim)
- parameters: the solutions of variables that were optimized
- objectives: non-dominated objective function values
- paretoFrontRank: nondomination ranks (or levels) that each non-dominated solution belongs to
- crowdingDistance: crowding distances of each non-dominated solution

Examples

```
## Not run:
source_names <- c(lr_network$source, sig_network$source, gr_network$source) %>% unique()
n_param <- length(source_names) + 4
n_obj <- 4
lower_bounds <- c(rep(0, times = length(source_names)), 0, 0, 0.9, 0.01)
upper_bounds <- c(rep(1, times = length(source_names)), 1, 1, 0.999, 0.99)
results <- run_nsga2R_cluster(model_evaluation_optimization_nsga2r,
```

```
  varNo = n_param, objDim = n_obj,
 lowerBounds = lower_bounds, upperBounds = upper_bounds, popSize = 360, tourSize = 2, generations = 15, ncores = 8
)

## End(Not run)
```

---

scale_quantile *Cut off outer quantiles and rescale to a [0, 1] range*

Description

scale_quantile Cut off outer quantiles and rescale to a [0, 1] range

Usage

```
scale_quantile(x, outlier_cutoff = .05)
```

Arguments

x	A numeric vector, matrix or data frame.
outlier_cutoff	The quantile cutoff for outliers (default 0.05).

Value

The centered, scaled matrix or vector. The numeric centering and scalings used are returned as attributes.

Examples

```
## Not run:
## Generate a matrix from a normal distribution
## with a large standard deviation, centered at c(5, 5)
x <- matrix(rnorm(200 * 2, sd = 10, mean = 5), ncol = 2)

## Scale the dataset between [0,1]
x_scaled <- scale_quantile(x)

## Show ranges of each column
apply(x_scaled, 2, range)

## End(Not run)
```

scale_quantile_adapted *Normalize values in a vector by quantile scaling and add a pseudovalue of 0.001*

Description

scale_quantile_adapted Normalize values in a vector by quantile scaling. Add a pseudovalue of 0.001 to avoid having a score of 0 for the lowest value.

Usage

```
scale_quantile_adapted(x, outlier_cutoff = 0)
```

Arguments

x	A numeric vector.
outlier_cutoff	The quantile cutoff for outliers (default 0).

Value

A quantile-scaled numeric vector.

Examples

```
## Not run:
scale_quantile_adapted(rnorm(5))

## End(Not run)
```

scaling_modified_zscore *Normalize values in a vector by the modified z-score method.*

Description

scaling_modified_zscore Normalize values by the modified z-score method (uses median and median absolute deviation instead meean)

Usage

```
scaling_modified_zscore(x)
```

Arguments

x	A numeric vector.

Value

A numeric vector.

Examples

```
scaling_modified_zscore(rnorm(5))
```

---

scaling_zscore *Normalize values in a vector by the z-score method*

Description

`scaling_zscore` Normalize values in a vector by the z-score method.

Usage

```
scaling_zscore(x)
```

Arguments

x	A numeric vector.

Value

A numeric vector.

Examples

```
scaling_zscore(rnorm(5))
```

---

single_ligand_activity_score_classification *Assess how well cells' ligand activities predict a binary property of interest of cells.*

Description

`single_ligand_activity_score_classification` Evaluates classification performances: it assesses how well cells' ligand activities can predict a binary property of interest.

Usage

```
single_ligand_activity_score_classification(ligand_activities, scores_tbl)
```

Arguments

ligand_activities	Output from the function 'normalize_single_cell_ligand_activities'.
scores_tbl	a tibble indicating for every cell whether the property of interests holds TRUE or FALSE (columns: $cell: character vector with cell ids and $score: logical vector according to property of interest).

Value

A tibble giving for every ligand, the classification performance metrics giving information about the relation between its activity and the property of interest.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
cell_ids <- c("cell1", "cell2")
expression_scaled <- matrix(rnorm(length(genes) * 2, sd = 0.5, mean = 0.5), nrow = 2)
rownames(expression_scaled) <- cell_ids
colnames(expression_scaled) <- genes
ligand_activities <- predict_single_cell_ligand_activities(cell_ids = cell_ids, expression_scaled = expression_sc
normalized_ligand_activities <- normalize_single_cell_ligand_activities(ligand_activities)
cell_scores_tbl <- tibble(cell = cell_ids, score = c(TRUE, FALSE))
classification_analysis_output <- single_ligand_activity_score_classification(normalized_ligand_activities, cel

## End(Not run)
```

---

single_ligand_activity_score_regression

Perform a correlation and regression analysis between cells' ligand activities and property scores of interest

---

Description

single_ligand_activity_score_regression Performs a correlation and regression analysis between cells' ligand activities and property scores of interest.

Usage

```
single_ligand_activity_score_regression(ligand_activities, scores_tbl)
```

Arguments

ligand_activities	Output from the function 'normalize_single_cell_ligand_activities'.
scores_tbl	a tibble containing scores for every cell (columns: $cell and $score). The score should correspond to the property of interest

Value

A tibble giving for every ligand, the correlation/regression coefficients giving information about the relation between its activity and the property of interest.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- list("TNF", "BMP2", "IL4")
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands, ltf_cutoff = 0, al
potential_ligands <- c("TNF", "BMP2", "IL4")
genes <- c("SOCS2", "SOCS3", "IRF1", "ICAM1", "ID1", "ID2", "ID3")
cell_ids <- c("cell1", "cell2")
expression_scaled <- matrix(rnorm(length(genes) * 2, sd = 0.5, mean = 0.5), nrow = 2)
rownames(expression_scaled) <- cell_ids
colnames(expression_scaled) <- genes
ligand_activities <- predict_single_cell_ligand_activities(cell_ids = cell_ids, expression_scaled = expression_sc
normalized_ligand_activities <- normalize_single_cell_ligand_activities(ligand_activities)
cell_scores_tbl <- tibble(cell = cell_ids, score = c(1, 4))
regression_analysis_output <- single_ligand_activity_score_regression(normalized_ligand_activities, cell_scores

## End(Not run)
```

---

source_weights_df *Data source weights*

Description

A data.frame/tibble describing weight associated to each data source. These weights will be used for weighted aggregagion of source networks

Usage

```
source_weights_df
```

Format

A data frame/tibble

source name of data source

weight weight associated to a data source

visualize_parameter_values *Visualize parameter values from the output of* run_nsga2R_cluster.

Description

visualize_parameter_values will take as input the output of run_nsga2R_cluster and visualize the data source weights and hyperparameters of the best and worst solutions

Usage

```
visualize_parameter_values(result_nsga2r, source_names, top_ns = c(5, 10, -10, -25))
```

Arguments

result_nsga2r	The output of run_nsga2R_cluster.
source_names	Character vector containing the names of the data sources.
top_ns	Numeric vector indicating how many of the best and worst solutions should be considered (negative values indicate the worst solutions; default: c(5, 10, -10, -25)).

Value

A list containing two ggplot objects, one for the data source weights and one for the hyperparameters.

Examples

```
## Not run:
results <- run_nsga2R_cluster(model_evaluation_optimization_nsga2r,
  varNo = n_param, objDim = n_obj,
 lowerBounds = lower_bounds, upperBounds = upper_bounds, popSize = 360, tourSize = 2, generations = 15, ncores = 8
)

# Visualize the best and worst 5 solutions
visualize_parameter_values(results, source_names, top_ns = c(5, -5))

## End(Not run)
```

visualize_parameter_values_across_folds *Visualize parameter values from the output of* run_nsga2R_cluster *across cross-validation folds.*

Description

visualize_parameter_values_across_folds will take as input the output of run_nsga2R_cluster and visualize the data source weights and hyperparameters of the best solutions across all folds.

Usage

```
visualize_parameter_values_across_folds(result_nsga2r_list, source_names, top_n = 25)
```

Arguments

result_nsga2r_list	A list containing the outputs of run_nsga2R_cluster for each cross-validation fold.
source_names	Character vector containing the names of the data sources.
top_n	Numeric indicating how many of the best solutions should be considered.

Value

A list containing two ggplot objects, one for the data source weights and one for the hyperparameters.

Examples

```
## Not run:
results_list <- lapply(cv_folds, function(fold) {
  settings <- readRDS(paste0("settings_training_f", fold))$settings
  forbidden_gr <- bind_rows(
   gr_network %>% filter(database == "NicheNet_LT" & from %in% settings$forbidden_ligands_nichenet),
   gr_network %>% filter(database == "CytoSig" & from %in% settings$forbidden_ligands_cytosig)
  )
  gr_network_subset <- gr_network %>% setdiff(forbidden_gr)
  run_nsga2R_cluster(model_evaluation_optimization_nsga2r,
    varNo = n_param, objDim = n_obj,
   lowerBounds = lower_bounds, upperBounds = upper_bounds, popSize = 360, tourSize = 2, generations = 15, ncores = 8
   source_names = source_names, algorithm = "PPR", correct_topology = FALSE, lr_network = lr_network, sig_network =
   settings = settings, secondary_targets = FALSE, remove_direct_links = "no", damping_factor = NULL
  )
})

# Visualize the best 25 solutions across all folds
visualize_parameter_values_across_folds(results_list, source_names, top_n = 25)
```

```
## End(Not run)
```

`wrapper_average_performances` *Calculate average performance of datasets of a specific ligand.*

Description

`wrapper_average_performances` Calculate average performance of datasets of a specific ligand. Datasets profiling more than one ligand (and thus ligands other than the ligand of interest), will be included as well.

Usage

```
wrapper_average_performances(ligand_oi,performances, averaging = "median")
```

Arguments

`ligand_oi`	Name of the ligand for which datasets should be averaged.
`performances`	A data frame with performance values, containing at least folowing variables: $setting, $ligand and one or more metrics.
`averaging`	How should performances of datasets of the same ligand be averaged? 'median' or 'mean'.

Value

A data frame containing classification evaluation measures for the ligand activity state prediction single, individual feature importance measures.

Examples

```
## Not run:
weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
ligands <- extract_ligands_from_settings(settings)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
perf1 <- lapply(settings, evaluate_target_prediction, ligand_target_matrix)
performances_target_prediction_averaged <- ligands %>%
  lapply(wrapper_average_performances, perf1, "median") %>%
  bind_rows() %>%
  drop_na()

## End(Not run)
```

wrapper_evaluate_single_importances_ligand_prediction *Evaluation of ligand activity prediction performance of single ligand importance scores: each dataset individually.*

Description

wrapper_evaluate_single_importances_ligand_prediction Evaluate how well a single ligand importance score is able to predict the true activity state of a ligand. For this it is assumed, that ligand importance measures for truely active ligands will be higher than for non-active ligands. Several classification evaluation metrics for the prediction are calculated and variable importance scores can be extracted to rank the different importance measures in order of importance for ligand activity state prediction.

Usage

```
wrapper_evaluate_single_importances_ligand_prediction(group,ligand_importances)
```

Arguments

group	Name of the dataset (setting) you want to calculate ligand activity performance for.
ligand_importances	A data frame containing at least folowing variables: $setting, $test_ligand, $ligand and one or more feature importance scores. $test_ligand denotes the name of a possibly active ligand, $ligand the name of the truely active ligand.

Value

A data frame containing classification evaluation measures for the ligand activity state prediction single, individual feature importance measures.

Examples

```
## Not run:
settings <- lapply(expression_settings_validation[1:5], convert_expression_settings_evaluation)
settings_ligand_pred <- convert_settings_ligand_prediction(settings, all_ligands = unlist(extract_ligands_from_s

weighted_networks <- construct_weighted_networks(lr_network, sig_network, gr_network, source_weights_df)
ligands <- extract_ligands_from_settings(settings_ligand_pred, combination = FALSE)
ligand_target_matrix <- construct_ligand_target_matrix(weighted_networks, lr_network, ligands)
ligand_importances <- dplyr::bind_rows(lapply(settings_ligand_pred, get_single_ligand_importances, ligand_target
evaluation <- ligand_importances$setting %>%
  unique() %>%
  lapply(function(x) {
    x
  }) %>%
  lapply(wrapper_evaluate_single_importances_ligand_prediction, ligand_importances) %>%
```

```
  bind_rows() %>%
  inner_join(ligand_importances %>% distinct(setting, ligand))
print(head(evaluation))

## End(Not run)
```

Index